LoRaWAN® Relay Specification TS011-1.0.0

NOTICE OF USE AND DISCLOSURE

LoRa Alliance®
5177 Brandin Court
Fremont, CA 94538
United States

LoRaWAN® Relay Specification TS011-1.0.0

Authored by the Relay/D2D Task Force of the LoRa Alliance Technical Committee

Technical Committee Chair and Vice-Chair:
A. YEGIN (Actility), O. SELLER (Semtech)

Relay/D2D Task Force Chairs:
D. ORIFIAMMA (Semtech), X. YU (Allibaba)

Editor:
B. MAI THANH (Semtech)

Contributors:
D. BARTHEL (Orange), C. BUFFARD (Kudelski IoT), I. CARRION (Charter), J. CATALANO (Kerlink), L. FERREIRA (Orange), K. GROCHLA (IITIS), M. GRUDSKY (Semtech), A. KASTTET (Birdz), M. KIRWAN (LoRa Alliance), D. KJENDAL (Senet), M. LUIS (Semtech), B. MAI THANH (Semtech), B. NING (Semtech), D. ORIFIAMMA (Semtech), O. SELLER (Semtech), A. YEGIN (Actility), X. YU (Allibaba)

Version: 1.0.0
Date: September 2022
Status: Released

Contents

1 Conventions ... 7
2 Introduction ... 8
3 Principle of Operation ... 9
3.1 Overview ... 9
3.2 Wake On Radio (WOR) ... 11
3.2.1 Channel Scan ... 11
3.2.2 Default Channel ... 13
3.2.3 Second Channel ... 13
3.2.4 WOR Reception ... 13
3.3 Wake On Radio Acknowledge (WOR ACK) ... 14
3.4 LoRaWAN Uplink ... 14
3.5 Message Forwarding ... 15
3.6 RXR Window ... 15
3.7 WOR Timing ... 15
3.7.1 WOR Relay Join-Request Timing ... 15
3.7.2 WOR Relay Class A Uplink Timing ... 15
3.8 Message Sequence Overview ... 17
3.9 Synchronization State ... 18
4 Security ... 20
4.1 Root Relay Session Key (`RootWorSKey`) ... 20
4.2 WOR Integrity Session Key (`WorSIntKey`) ... 20
4.3 WOR Encryption Session Key (`WorSEncKey`) ... 20
4.4 `RootWorSKey` Key Derivation ... 20
4.5 `WorSIntKey` and `WorSEncKey` Key Derivation ... 21
5 Wake On Radio Detail ... 22
5.1 Physical Parameters ... 22
5.2 Preamble Length ... 22
5.3 Payload Format ... 22
5.3.1 WOR Relay Join-Request ... 23
5.3.2 WOR Relay Class A Uplink ... 23
6 WOR ACK Detail ... 25
6.1 Physical Parameters ... 25
6.2 Payload Format ... 25
7 RXR Window ... 28
7.1 Physical Parameters ... 28
7.2 Window Opening Condition ... 28
8 Message Forwarding ... 29
8.1 Uplink Encapsulation ... 29
8.2 Uplink Forwarding Timing ... 29
8.3 Uplink Maximum Payload Size ... 29
8.4 Downlink Encapsulation ... 30
8.5 Downlink Maximum Payload Size ... 30
8.6 Forward/Filter Join-Request ... 30
8.7 Notify Standard Class A Uplink ... 31
8.8 Forwarding Limitations ... 31
9 Forwarded Message Protocol ... 33
9.1 Forward Uplink from an End-Device (*ForwardUplinkReq*) ... 33
9.2 Downlink to a Relay for Further Forwarding (*ForwardDownlinkReq*) ... 34
10 Relay MAC Commands ... 35
10.1 Update Relay Configuration (*RelayConfReq, RelayConfAns*) ... 35
10.2 Update End-Device Configuration (*EndDeviceConfReq, EndDeviceConfAns*) ... 37

10.3 Manage Join-Request Forward List (*FilterListReq*, *FilterListAns*) ... 39
10.4 Add End-Device to Uplink Forward List (*UpdateUplinkListReq*, *UpdateUplinkListAns*) ... 40
10.5 Manage End-Device from Uplink Forward List (*CtrlUplinkListReq*, *CtrlUplinkListAns*) ... 41
10.6 Manage Forwarding Limitations (*ConfigureFwdLimitReq*, *ConfigureFwdLimitAns*) ... 42
10.7 Notify New End-Device Under a Relay (*NotifyNewEndDeviceReq*) ... 44
11 Certification ... 45
11.1 Relay Certification ... 45
11.2 End-Device Certification ... 45
12 Glossary ... 46
13 Bibliography ... 47
13.1 References ... 47
APPENDIX 1: Relay/End-Device Synchronization ... 48
APPENDIX 2: Security Considerations ... 52
A. Relay Node and Associated Relay Protocol ... 52
B. Relay Protocol Between the End-Device and NS ... 53
C. WOR Protocol ... 53
D. New Links ... 55
APPENDIX 3: Filter/Forward Example ... 58
APPENDIX 4: Network Time Synchronization ... 59
APPENDIX 5: End-Device Relay Mode Activation Management ... 60

Tables

Table 1: RootWorSKey key source derivation ... 20
Table 2: WOR format ... 22
Table 3: WORHeader format ... 22
Table 4: WORType decoding ... 22
Table 5: WOR Relay Join-Request format ... 23
Table 6: WorDrPL format ... 23
Table 7: WOR Relay Class A Uplink ... 23
Table 8: WorUplink format ... 23
Table 9: WorDrPL format ... 24
Table 10: Block A_{WOR} format ... 24
Table 11: B0 block for MIC computation in WOR Relay Class A Uplink ... 24
Table 12: WOR ACK Relay Class A Uplink format ... 25
Table 13: AckUplink format ... 25
Table 14:StateSync format ... 25
Table 15: CadToRx decoding ... 25
Table 16: Forward decoding ... 25
Table 17: XTALAccuracy decoding ... 26
Table 18: CADPeriodicity decoding ... 26
Table 19: Block A_{ACK} format ... 26
Table 20: WOR block for MIC computation in WOR ACK ... 27
Table 21: B0 block for MIC computation in WOR ACK ... 27
Table 22: Forwarded uplink timing ... 29
Table 23: Forwarding limit list ... 31
Table 24: Message limits ... 32
Table 25: Network session key vs LoRaWAN version ... 33
Table 26: ForwardUplinkReq format ... 33
Table 27: UplinkMetadata format ... 33

Table 28: WORChannel decoding ... 33
Table 29: ForwardDownlinkReq ... 34
Table 30: Summary of relay protocol command messages ... 35
Table 31: RelayConfReq format ... 35
Table 32: ChannelSettingsRelay format ... 36
Table 33: StartStop decoding ... 36
Table 34: Second channel decoding ... 36
Table 35: WOR ACK frequency offset ... 36
Table 36: RelayConfAns format ... 37
Table 37: Status format ... 37
Table 38: EndDeviceConfReq format ... 37
Table 39: ActivationRelayMode format ... 37
Table 40: RelayModeActivation format ... 37
Table 41: SmartEnableLevel format ... 38
Table 42: ChannelSettingsED format ... 38
Table 43: BackOff decoding ... 38
Table 44: EndDeviceConfAns format ... 38
Table 45: Status format ... 39
Table 46: FilterListReq format ... 39
Table 47: FilterListParam format ... 39
Table 48: FilterListAction decoding ... 39
Table 49: *FilterListAns* format ... 40
Table 50: Status format ... 40
Table 51: UpdateUplinkListReq format ... 40
Table 52: UplinkListIdxPL format ... 40
Table 53: UplinkLimitPL format ... 41
Table 54: UplinkLimitReloadRate decoding ... 41
Table 55: UplinkLimitBucketSize decoding ... 41
Table 56: CtrlUplinkListReq format ... 42
Table 57: CtrlUplinkActionPL format ... 42
Table 58: CtrlUplinkAction decoding ... 42
Table 59: CtrlUplinkListAns format ... 42
Table 60: Status format ... 42
Table 61: ConfigureFwdLimitReq format ... 43
Table 62: `FwdLimitReloadRatePL` format ... 43
Table 63: ResetLimitCounter decoding ... 43
Table 64: Forward Limit decoding ... 43
Table 65: FwdLimitLoadCapacityPL format ... 43
Table 66: NotifyNewEndDeviceReq format ... 44
Table 67: PowerLevel format ... 44

Figures

Figure 1: LoRaWAN relay mechanism principle ... 9
Figure 2: End-device under relay: WOR and RXR window ... 10
Figure 3: Single channel scan periodicity ... 11
Figure 4: Dual channel scan periodicity ... 12
Figure 5: Single channel scan interruption ... 12
Figure 6: Dual channel scan interruption ... 12
Figure 7: Switch CAD to RX ... 13
Figure 8: WOR Relay Join-Request timing ... 15
Figure 9: WOR Relay Class A Uplink timing ... 16

Figure 10: Missed WOR ACK timing 16
Figure 11: Full sequence diagram 17
Figure 12: New end-device under relay 17
Figure 13: Synchronization states 19
Figure 14: TOffset computation 26
Figure 15: Synchronization median and worst cases 48
Figure 16: Relayed network architecture 52
Figure 17: New radio links 55
Figure 18: Network time synchronization 59

1 Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in IETF Best Current Practice 14 (BCP14 [RFC2119] [RFC8174]) when, and only when, they appear in all capitals, as shown here.

The tables in this document are normative. The figures and notes in this document are informative.

Referenced document titles are written as *LoRaWAN Layer 2 Specification* and referenced section titles within this document are written as “Appendix 2: Security Considerations”.

Commands are written ***NewChannelReq***, bits and bit fields are written `CADPeriodicity`, constants are written RECEIVE_DELAY1, variables are written *N*.

In this document:

- The octet order for all multi-octet fields SHALL be little endian.
- EUI are 8-octet fields and SHALL be transmitted as little endian.
- By default, `RFU` bits are Reserved for Future Use and SHALL be set to 0 by the transmitter of the packet and SHALL be silently ignored by the receiver.

2 Introduction

This document describes the relaying mechanism that may be used to transport LoRaWAN frames bi-directionally between an end-device and a gateway/Network Server through a relay.

A relay is a LoRaWAN end-device based on the same hardware architecture as a regular end-device (typically a micro-controller, a radio and an antenna), and is often battery-powered. A relay can transfer LoRaWAN frames between an end-device and a LoRaWAN network in both directions (uplink and downlink), which may be useful when the end-device has insufficient coverage from LoRaWAN gateways to establish and maintain a satisfactory direct connection with the network.

To make this kind of communication possible, additional capabilities and controls are required, in addition to those defined by the existing *LoRaWAN Layer 2 Specification* [TS001]. This document specifies these additional capabilities and controls required by the end-device, the relay, and the Network Server.

3 Principle of Operation

3.1 Overview

The relay is a hardware device whose goal is to forward messages from an end-device to the network (and vice-versa) over the LoRaWAN air interface. To enable this new functionality, a new protocol has been developed.

This protocol includes a Wake On Radio (WOR) frame sent by the end-device to the relay. The goal of this WOR frame is to “wake up” the relay and communicate information about the LoRaWAN frame that will need to be forwarded.

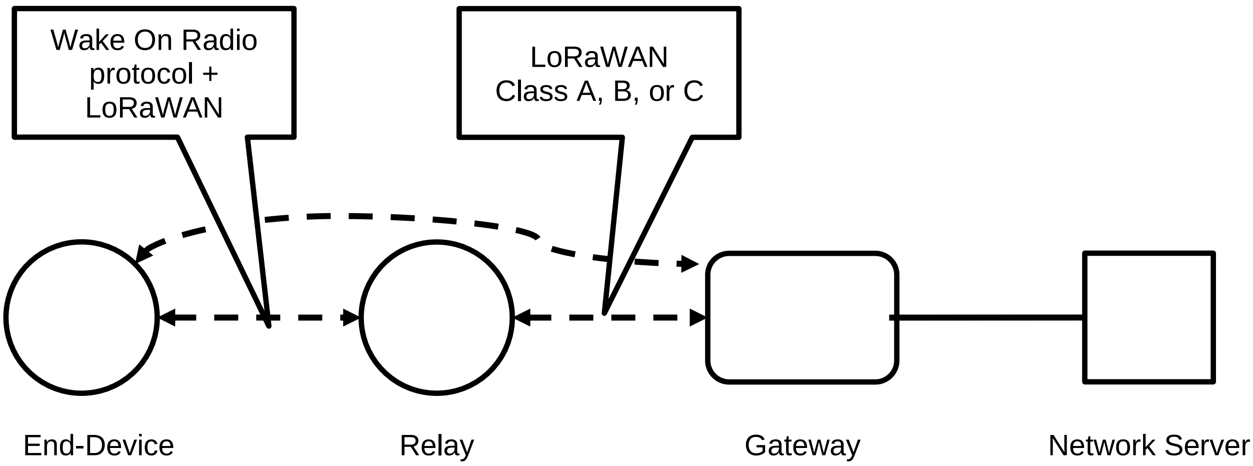

Figure 1: LoRaWAN relay mechanism principle

The main characteristic of the WOR frame is its preamble size, which can be up to 1 second. This long preamble allows the relay to sleep and only wake up periodically to scan for radio activity. This action is the channel scan.

If no radio activity is detected during a channel scan, the relay will go back to sleep and will try to detect a WOR frame during its next channel scan. However, if radio activity is detected, the relay will try to demodulate the message and check if it is a valid WOR frame.

If it is a valid WOR frame, the relay may reply with a Wake On Radio Acknowledgement (WOR ACK) frame. The purpose of this is to send information about the relay’s internal state and let the end-device know that its WOR frame has been received. Now, the end-device is ready to send its LoRaWAN uplink. This uplink will be received by the relay and then forwarded to the Network Server using the relay’s own LoRaWAN link.

The relay protocol is meant to be developed on top of the LoRaWAN link layer specification. As of the publication date of this document, all of the currently existing versions of LoRaWAN will support this protocol, both from the end-device and from the relay.

A new reception window (named *RXR*) has been created to allow an end-device under a relay to receive forwarded downlinks. Figure 2 shows this process:

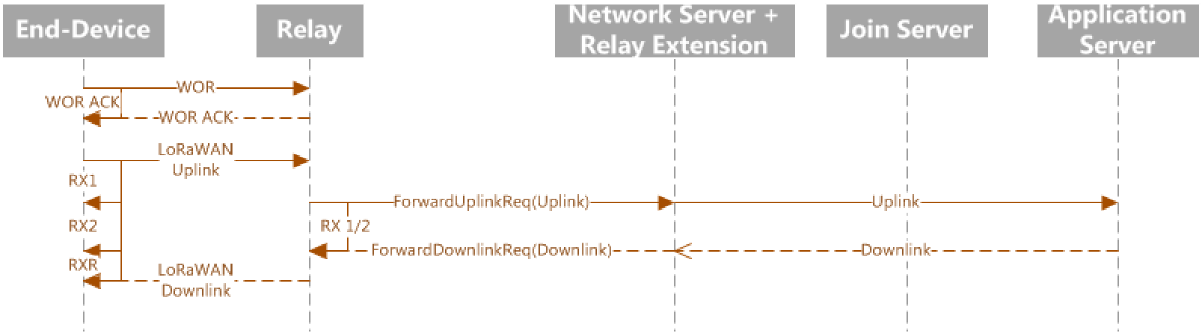

Figure 2: End-device under relay: WOR and RXR window

Prior to joining a LoRaWAN network, an end-device SHOULD autonomously decide to enable or not enable the relay mode. Refer to APPENDIX 5 for recommendations on how an end-device should manage the relay mode.

3.2 Wake On Radio (WOR)

All communication between the end-device and the relay SHALL be initiated by the end-device.

The end-device first sends a WOR frame. The purpose of this WOR frame is to wake-up the relay and signal to the relay the radio parameters that will be used for the subsequent LoRaWAN frame.

The WOR frame preamble can be up to 1 second long. Its length depends on the synchronization state between the end-device and the relay. This long preamble is sent by the end-device to allow the relay to only listen for a short duration, periodically, instead of listening continuously.

There are two types of WOR frames used to communicate through a relay: one is dedicated to the LoRaWAN join procedure and the other one is for standard class A uplinks.

3.2.1 Channel Scan

To detect the WOR frame, the relay periodically performs a Channel Activity Detection (CAD) to detect radio activity. This scan is done on the channel(s) supported by the relay. A relay SHALL support at least one channel, which is referred to as the *default channel*.

The time between the beginning of two consecutive channel scans is defined by the parameter `CADPeriodicity`[1]. The default `CADPeriodicity` of the default channel is 1000ms. It MAY have a lower value.

The following diagram illustrates the scanning pattern when a single channel is used.

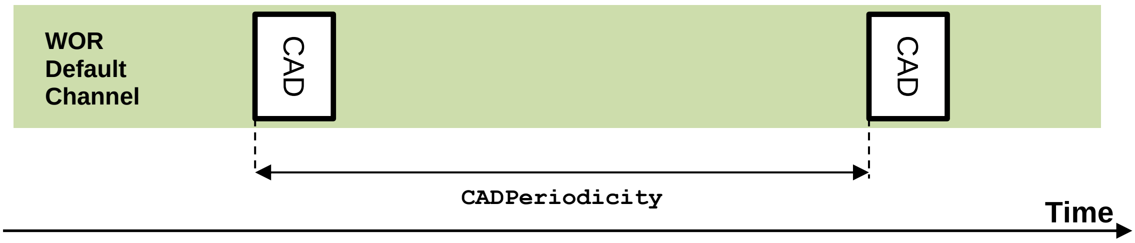

Figure 3: Single channel scan periodicity

A relay MAY support a second channel. When two channels are used, the relay alternately scans the two channels. In that case, the delay between two consecutive scans is `CADPeriodicity`/2; therefore two consecutive scans on the same channel occur every `CADPeriodicity`.

[1] See `CADPeriodicity` in section 6.2 for the possible values.

The following diagram illustrates the scanning pattern when two channels are used:

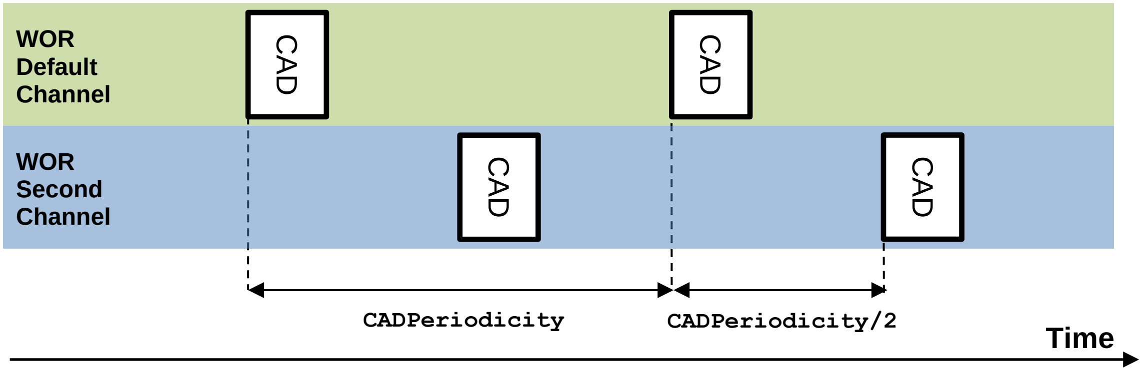

Figure 4: Dual channel scan periodicity

If a relay is unable to scan the WOR channel(s) for some period of time, it SHALL resume its scan on the next integer multiple of the `CADPeriodicity` interval as shown in Figure 5.

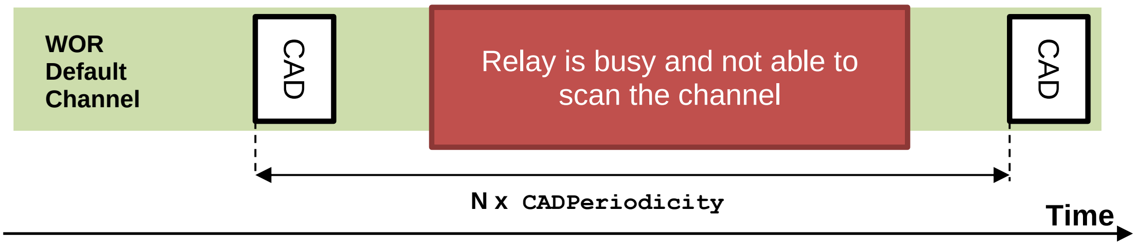

Figure 5: Single channel scan interruption

If the second channel is enabled, the relay SHALL scan the second channel on the next integer multiple of `CADPeriodicity` plus half the `CADPeriodicity` `(CADPeriodicity*(N+1/2))`.

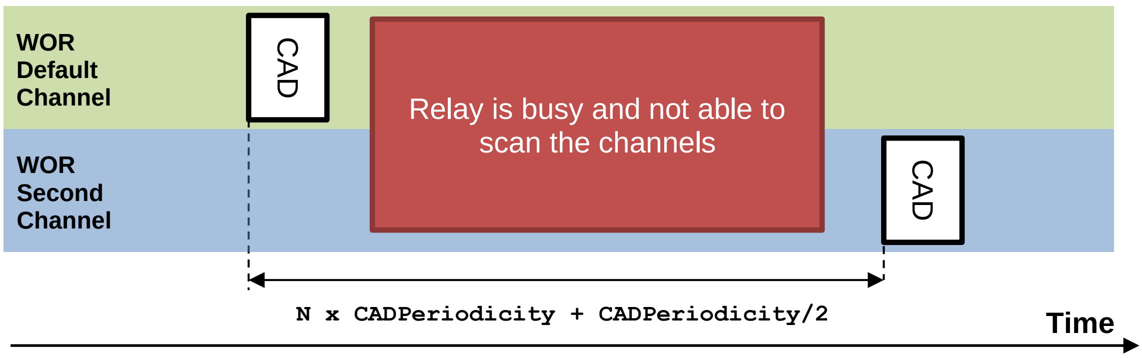

Figure 6: Dual channel scan interruption

3.2.2 Default Channel

Refer to [RP002] for the definitions of the physical parameters of this channel.

The default channel of any given relay can be one of up to two different configurations. The end-device SHALL alternate WOR transmissions with each defined configuration of the default channel until it receives a valid downlink on the RX relay window[2] or a valid WOR ACK[3].

3.2.3 Second Channel

The Network Server can create a new channel (referred to as *second channel*) and can configures the parameters of this channel on both the relay and the end-device using Medium Access Control (MAC) commands[4].

> **Note:** The Network Server will be able to send this channel configuration to the end-device only when the latter is connected to the network. This channel may use a different frequency and/or data rate than the default channel to improve communication speed and/or frequency diversity.

The end-device SHOULD preferably use the second channel (if one is defined).

3.2.4 WOR Reception

If radio activity is detected during a scan, the relay SHOULD attempt to demodulate the incoming packet.

The duration of the CAD is not null, and the relay may require some time to switch from CAD to reception mode. This delay is called `CadToRx` and SHALL be communicated to the end-device using the WOR ACK frame.

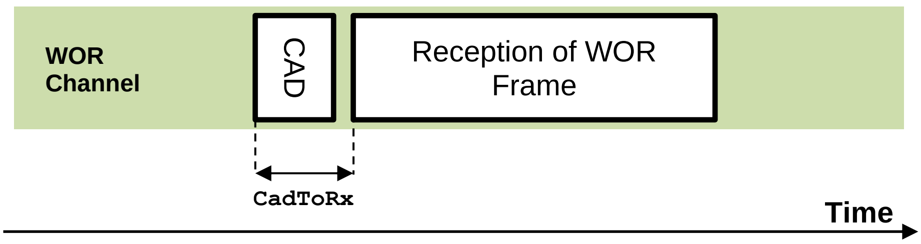

Figure 7: Switch CAD to RX

Once a valid WOR frame has been received, the relay MAY proceed to:

- Send a WOR ACK (only for WOR Class A Uplink)
- Listen to the subsequent LoRaWAN message

> **Note:** The relay may implement protection mechanisms to maximize its battery life. For example, the relay may decide to drop packets whose preamble exceed the CAD periodicity, therefore avoiding demodulating

[2] Refer to Section 3.6 RXR Window
[3] Refer to Section 3.3 Wake On Radio Acknowledge (WOR ACK)
[4] Refe3r to Section 10.1 Update Relay Configuration (*RelayConfReq, RelayConfAns*)

> packets that cannot be legitimate WOR frames. The implementation of those filtering techniques is not part of the relay specification.

A relay SHALL NOT attempt to demodulate the subsequent LoRaWAN uplink (Join-Request or Class A Uplink) if:

- The Message Integrity Code (MIC) of the WOR frame was determined to be incorrect (See 4.2), or
- The forwarding limit has been reached (See 8.8), or
- The Join-Request `JoinEUI` / `DevEUI` is filtered (See 8.6).

The relay SHALL be able to check the MIC validity of TRUSTED_END_DEVICES_NUMBER end-devices. Each of these end-devices is referred to as a “trusted end-device”. Refer to [RP002] for more information.

3.3 Wake On Radio Acknowledge (WOR ACK)

The relay MAY send a WOR ACK frame to the end-device. The relay SHALL NOT send a WOR ACK frame if the WOR frame is invalid.

The purposes of the WOR ACK frame are:

- Time synchronization (allowing the end-device to use a shorter preamble for the next messages) (See 3.9).
- Establish a trusted communication between the end-device and the relay (See 4).
- Provide relay forwarding information to the end-device (See 8.8).

A relay SHALL NOT acknowledge a WOR Relay Join-Request frame.

> **Note:** Only WOR Standard Class A Uplinks can be authenticated (since the relay session key is derived after a Join-Request) and require a WOR ACK frame.

3.4 LoRaWAN Uplink

Once the end-device has sent a WOR frame and received (or not) a WOR ACK frame, it MAY send its LoRaWAN uplink.

> **Note:** A gateway within range is also able to receive this uplink as it is a standard LoRaWAN uplink.

In case the end-device was waiting for a WOR ACK frame and didn’t receive one, it could:

- Send its uplink anyway.
- Retry sending the WOR frame up to *X* times before sending its uplink.

By default, an end-device SHOULD send its LoRaWAN uplink after each WOR_ATTEMPTS_WO_ACK WOR attempt even without having received a WOR ACK frame. Refer to [RP002] for more information.

The Network Server MAY send a MAC command to change this behavior. Refer to the `BackOff` field in Section 10.2, “Update End-Device Configuration (*EndDeviceConfReq, EndDeviceConfAns*)”.

It is RECOMMENDED to configure the end-device to send its uplink every so often even without receiving the WOR ACK frame, in order to increase resilience to an intermittent Denial-of-Service (DoS) attack again the WOR link. For more information refer to “APPENDIX 2: Security Considerations”.

The relay SHALL stop to demodulate the incoming message if it is too long, regarding the maximum payload size authorized at this data rate.

3.5 Message Forwarding

Once a relay receives an end-device LoRaWAN uplink, it SHOULD forward it to the Network Server.

If the Network Server wants to send a downlink to the end-device, it SHALL reply on the RX1 or RX2 windows of the relay.

3.6 RXR Window

If the Network Server replies to a forwarded uplink and the relay decides to forward it to the end-device, the relay SHALL send this downlink to the end-device on the end-device’s RXR (RX Relay) window. For more information, refer to Section 7.

> **Note:** An end-device is still able to receive a downlink on the RX1 or RX2 windows directly from a gateway.

3.7 WOR Timing

3.7.1 WOR Relay Join-Request Timing

The delay between the end of the WOR frame and the beginning of the LoRaWAN Join-Request is constant and defined as WOR_DATA_DELAY (see Figure 8). Refer to [RP002] for more information.

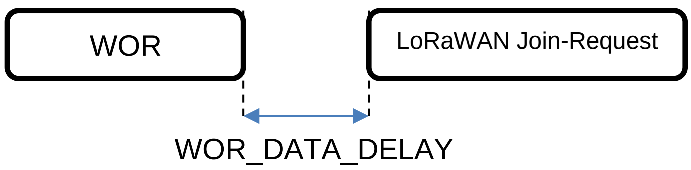

Figure 8: WOR Relay Join-Request timing

3.7.2 WOR Relay Class A Uplink Timing

A relay MAY acknowledge with a WOR ACK frame for any valid WOR Relay Class A Uplink it receives.

The delay between the end of the WOR frame and the beginning of the WOR ACK frame is constant and defined as WOR_ACK_DELAY. The delay between the end of the WOR ACK frame and the beginning of the LoRaWAN uplink is constant and equal to WOR_DATA_DELAY. Refer to [RP002] for more information.

Figure 9 represents the timing when the end-device receives the WOR ACK frame.

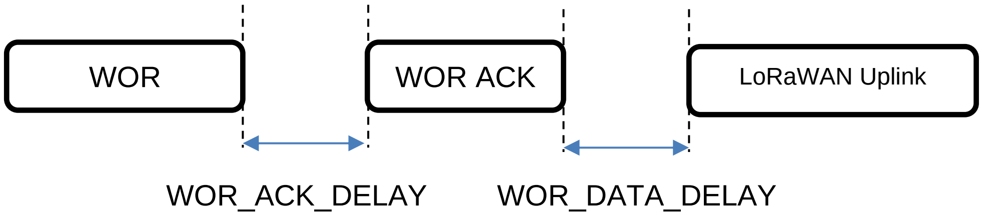

Figure 9: WOR Relay Class A Uplink timing

In case the end-device decides to send its LoRaWAN uplink without having received a WOR ACK frame, the end-device SHALL schedule the transmission as if the WOR ACK frame was present.

In this case, the delay between the end of the WOR frame and the beginning of the LoRaWAN uplink is constant and defined as WOR_ACK_DELAY + WOR_ DATA_DELAY + TimeOnAir (WOR_ACK), as shown on Figure 10.

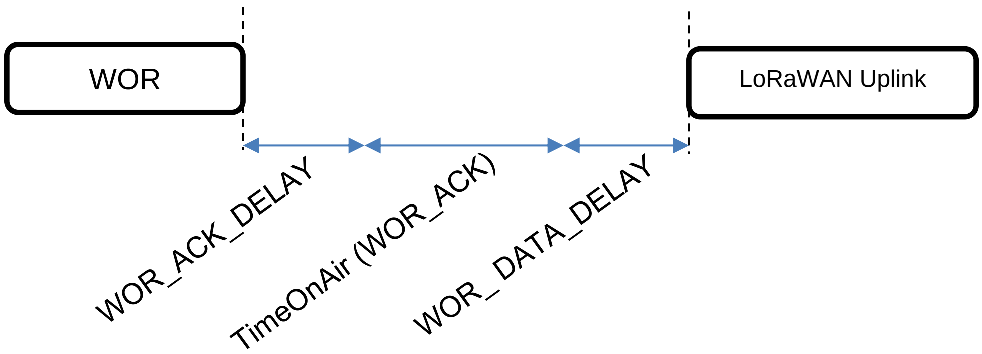

Figure 10: Missed WOR ACK timing

TimeOnAir (WOR_ACK) is dependent on the data rate and should be computed by the end-device. Refer to [RP002] for the possible values.

3.8 Message Sequence Overview

Figure 11 shows how an end-device can join and send data through a relay.

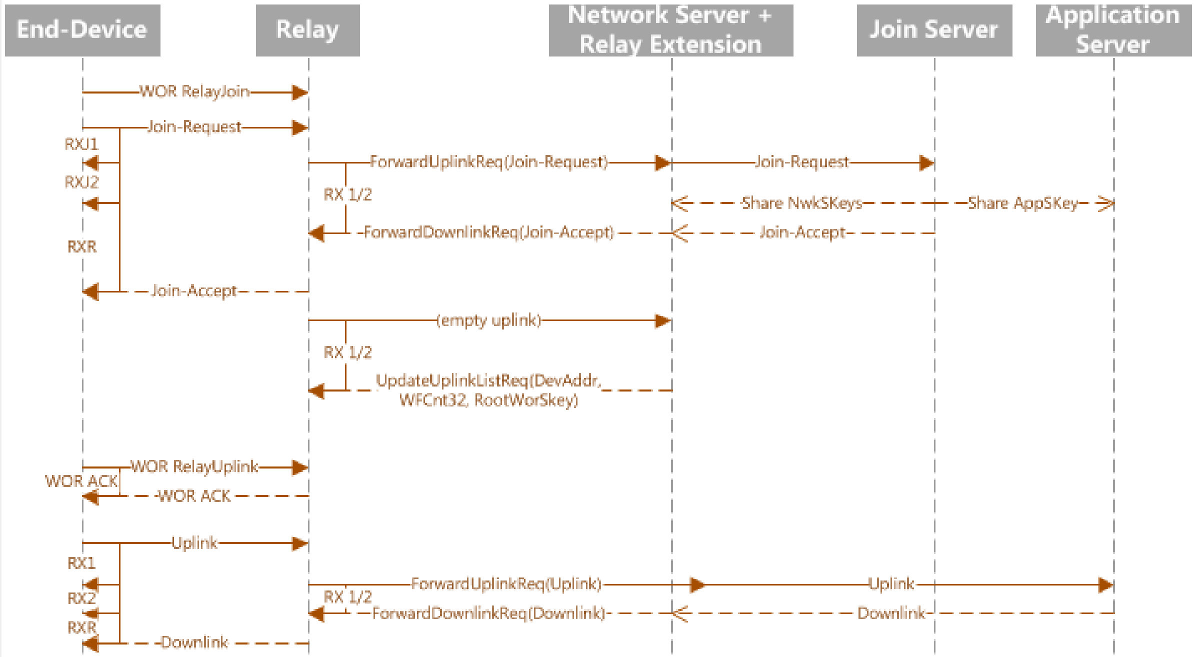

Figure 11: Full sequence diagram

Figure 12 shows how a relay behaves when it receives WOR frames from:

- An end-device that has already (re)joined the network directly through a gateway or under a different relay.
- An Activation By Personalization (ABP) end-device.

In both cases, the relay MAY notify the Network Server that a new end-device has been detected within its range. The Network Server MAY respond by sending the relay session key (`RootWorSKey`) for the relayed end-device. Figure 12 shows the case when the Network Server responds with the relay session key.

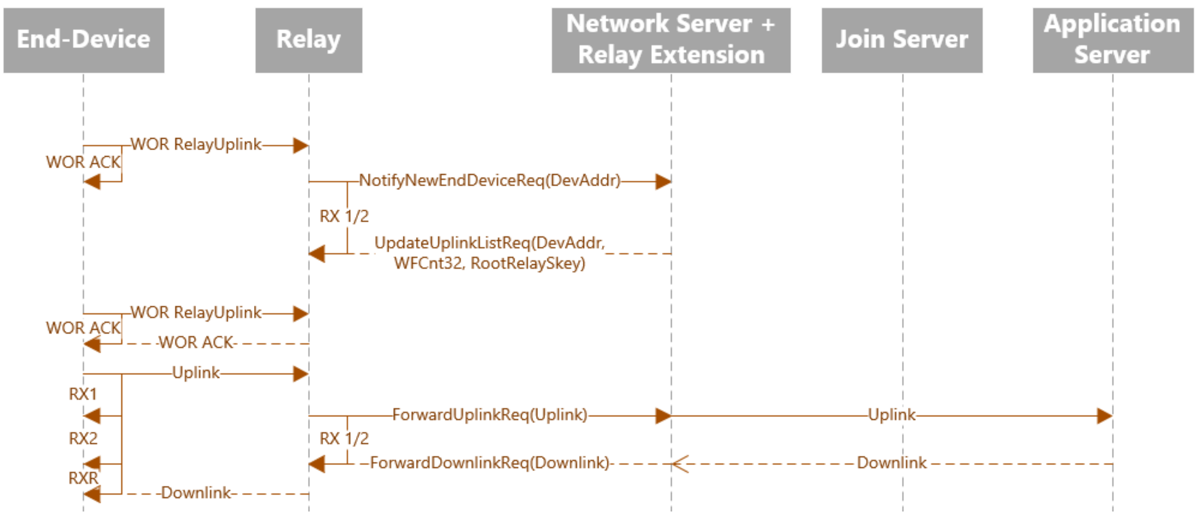

Figure 12: New end-device under relay

3.9 Synchronization State

Regarding its synchronization with a relay, an end-device can be in three possible states: **initialized**, **unsynchronized,** or **synchronized**.

In the **initialized** state, the end-device is unaware of the following variables:

- Exact time at which the relay will be listening on the default channel
- Relay CAD periodicity
- Relay crystal oscillator frequency accuracy
- Presence (or not) of the second channel
- Delay between CAD to RX mode of the relay
- Channel index of the default channel

An end-device in the **initialized** state SHALL assume the following values:

- No second channel
- CAD periodicity of 1000 milliseconds (ms)
- Crystal accuracy of 40 parts per million (ppm)
- 8 symbols to switch from CAD to RX

If multiple configurations are possible for the default channel, the end-device SHALL alternate between possible configurations until it receives a valid WOR ACK frame or a valid downlink on the RXR.

An end-device in **initialized** or **unsynchronized** state MAY transmit a WOR frame whenever it wants. The preamble size of this WOR frame SHOULD be set to the maximum value, i.e., 1000 ms.

When the end-device receives a WOR ACK frame, it SHALL move to the **synchronized** state. In this state, the end-device knows the CAD periodicity, the delay to switch from CAD to RX, and the crystal accuracy of the relay. It is also able to estimate when the relay scans the channel(s) and can therefore reduce the preamble length of its next WOR frames[5].

An end-device SHALL move from **initialized** to **unsynchronized** after a join if it has received the Join-Accept in the RXR window.

Due to crystal frequency inaccuracy, the window estimated by the end-device for the relay scan event widens over time. When this window is greater than the CAD periodicity, the end-device SHALL move to the **unsynchronized** state.

When a **synchronized** end-device has sent 8 consecutive WOR Class A Uplink frames without receiving a WOR ACK frame, it SHALL switch to the **unsynchronized** state.

When an **unsynchronized** end-device has sent 8 consecutive WOR Class A Uplink frames without receiving a WOR ACK frame, it SHALL switch to the **initialized** state and discard the current configuration of the relay.

The following diagram illustrates the three possible synchronization states that an end-device may be in:

[5] Refer to “APPENDIX 1: Relay/End-Device Synchronization”

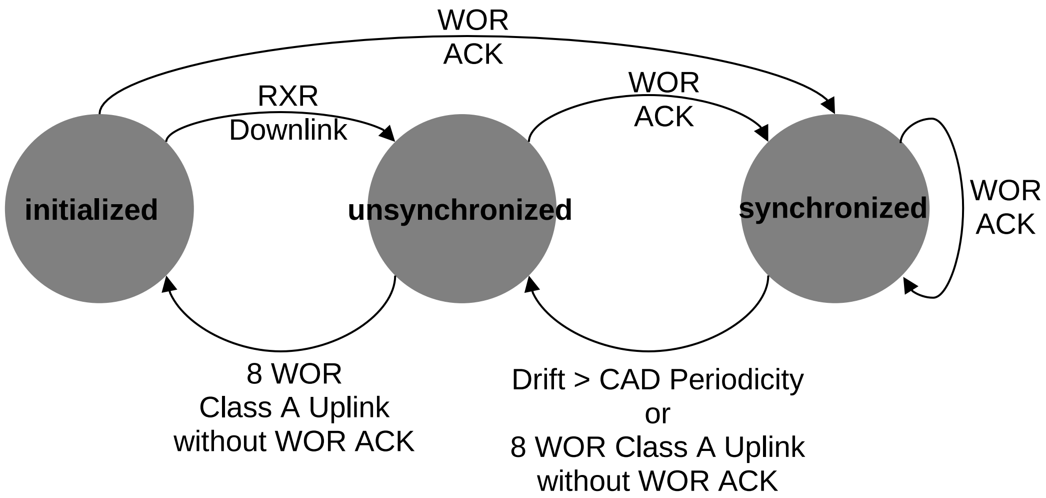

Figure 13: Synchronization states

> **Note:** A Network Server is not notified when the synchronization state of the end-device changes but it can try to guess if the end-device is in the **initialized** state because the WOR channel used (default or second) will be specified in the forwarded message. If it thinks that the end-device is in the **initialized** state, the Network Server can send the second channel configuration.

4 Security

4.1 Root Relay Session Key (`RootWorSKey`)

The `RootWorSKey` is a relay session key specific to the end-device. It is used by both the end-device and the relay to derive the `WorSIntKey` and `WorSEncKey`.

For OTAA end-devices, this key SHALL be derived after receiving a valid Join-Accept frame.

For ABP end-devices, this key SHALL be saved in non-volatile memory and provisioned into the end-device together with the two session keys, `NwkSKey` and `AppSKey`.

The `RootWorSKey` is never used directly to compute the MIC or to encrypt/decrypt data.

> **Note:** The purpose of this intermediate key is to minimize the payload size from the Network Server to the relay when sending end-device relay credentials to a relay.

4.2 WOR Integrity Session Key (`WorSIntKey`)

`WorSIntKey` is a security session key specific to the end-device. It is used by both the relay and the end-device to calculate and verify the MIC of the WOR and WOR ACK frames to ensure data integrity.

4.3 WOR Encryption Session Key (`WorSEncKey`)

`WorSEncKey` is a security session key specific to the end-device. It is used by both the relay and the end-device to encrypt and decrypt the payload field of all WOR and WOR ACK frames.

4.4 `RootWorSKey` Key Derivation

For Over-The-Air Activation (OTAA) end-devices, the `RootWorSKey` SHALL be calculated as follows:

$$\texttt{RootWorSKey} = \text{aes128\_encrypt}(\texttt{Key},\ \texttt{0x01} \mid \text{pad}_{16})$$

With:

- `Key`: An AES-128 session key specific to the end-device. **Error! Reference source not found.** shows the key to use depending on the end-device LoRaWAN version. The key name is the same as the one defined in [TS001].

LoRaWAN Version	Key to Use
1.0.X	`NwkSKey`
1.1.X and higher	`NwkSEncKey`

Table 1: RootWorSKey key source derivation

4.5 `WorSIntKey` and `WorSEncKey` Key Derivation

The `WorSIntKey` SHALL be calculated as follows:

`WorSIntKey` = aes128_encrypt(`RootWorSKey, 0x01 | DevAddr |` pad_{16})

The `WorSEncKey` SHALL be calculated as follows:

`WorSEncKey` = aes128_encrypt(`RootWorSKey, 0x02 | DevAddr |` pad_{16})

5 Wake On Radio Detail

5.1 Physical Parameters

Refer to [RP002] for more information on the physical parameters.

5.2 Preamble Length

The preamble length is variable and depends on the synchronization state between the end-device and the relay. Refer to "APPENDIX 1: Relay/End-Device Synchronization" for more information on the formula to compute the preamble length.

For end-devices in the **initialized** or **unsynchronized** states, the preamble length is given by the following formula:

$$PreambleLength_{SYMB} = floor\left(\frac{CADPeriodicity}{T_{SYMB}}\right) + 1 + 6 + \mathrm{CadToRx}_{\mathrm{SYMB}}$$

`CadToRx` is information received in the WOR ACK frame. In the **initialized** state, an end-device SHALL assume the highest possible value for `CadToRx`. Refer to "Table 15: CadToRx decoding" for the possible values.

> **Note:** The 6 symbols shown in the formula above are the minimum required by the relay radio transceiver to demodulate an incoming message.

5.3 Payload Format

The WOR frame has the following format:

Size (octets)	1	variable
`WORFrame`	`WORHeader`	`WORPayload`

Table 2: WOR format

With `WORHeader` containing:

`Bits`	`7:4`	`3:0`
`WORHeader`	`RFU`	`WORType`

Table 3: WORHeader format

The `WORType` is described below:

`WORType`	**Description**
`0`	Relay Join-Request
`1`	Relay Class A Uplink
`2..14`	RFU
`15`	Proprietary

Table 4: WORType decoding

The `WORPayload` format depends of the `WORType` field. The possible formats for `WORType` 0 and 1 are described in the next sections.

5.3.1 WOR Relay Join-Request

For WOR Relay Join-Request, `WORPayload` is defined in Table 5.

Size (octets)	1	3
WORPayload	WorDrPL	Frequency

Table 5: WOR Relay Join-Request format

Where:

Bits	7:4	3:0
WorDrPL	RFU	DataRate

Table 6: WorDrPL format

The `DataRate` field defines the data rate used by the next Join-Request uplink frame. Refer to [RP002] for more information on the bandwidth and spreading factor.

The `Frequency` field corresponds to the frequency used by the next Join-Request uplink frame, whereby the frequency is coded following the convention defined in the [TS001] ***NewChannelReq*** command.

5.3.2 WOR Relay Class A Uplink

For WOR Relay Class A Uplinks, `WORPayload` is defined in Table 7.

Size (octets)	4	4	2	4
WORPayload	DevAddr	WorUplinkEnc	WFCnt	MIC

Table 7: WOR Relay Class A Uplink

`DevAddr` is the 32-bit device address of the end-device as defined in [TS001].

The frame counter (`WFCnt32`) is 32 bits wide. The `WFCnt` field SHALL correspond to the least-significant 16 bits of the 32-bit frame counter.

The end-device SHALL NOT reuse the same `WFCnt32` value with the same relay session keys. Each time the end-device sends a new WOR frame, it SHALL increment the `WFCnt32` by 1.

Whenever an OTAA end-device successfully processes a Join-Accept frame, `WFCnt32` SHALL be reset to `0`.

For ABP end-devices, the WOR frame counter is initialized to `0` by the manufacturer. ABP end-devices SHALL NOT reset the WOR frame counter during the end-device's lifetime. If the end-device is susceptible to losing power during its lifetime (e.g., battery replacement), the WOR frame counter SHALL persist during the event.

The field `WorUplinkEnc` is the result of encrypting the `WorUplink` field with `WorSEncKey`. This field conveys the physical parameters of the subsequent LoRaWAN frame (see Table 8 and Table 9).

Size (octets)	1	3
WorUplink	WorDrPL	Frequency

Table 8: WorUplink format

Bits	7:4	3:0
WorDrPL	RFU	DataRate

Table 9: WorDrPL format

The `DataRate` field defines the data rate used by the corresponding subsequent uplink frame. Refer to [RP002] for more information on the bandwidth and spreading factor.

The `Frequency` field corresponds to the frequency used by the next uplink frame; it is encoded following the convention defined in the [TS001] ***NewChannelReq*** command.

`WorUplink` is encrypted as shown in Table 10; for each WOR Relay Class A Uplink frame, a block A_{WOR} SHALL be composed.

Size (octets)	1	2	1	4	4	3	1
A_{WOR}	0x01	2x 0x00	0 (Uplink)	DevAddr	WFcnt32	WOR Frequency	WOR Datarate

Table 10: Block A_{WOR} format

`WORFrequency` is the frequency used to send this WOR Relay Class A Uplink frame; it is encoded the same way as `Frequency`.

`WORDataRate` is the data rate used to send this WOR Relay Class A Uplink frame; it is encoded the same way as `Datarate` and has the same minimum and maximum value.

The block A_{WOR} SHALL be encrypted to obtain a block S_{WOR}.

S_{WOR} = aes128_encrypt(`WorSEncKey`, A_{WOR})

Encryption and decryption of the payload SHALL be calculated as follows:

`WorUplinkEncPad` = (`WorUplink` | pad_{16}) xor S_{WOR}
`WorUplinkEnc` = `WorUplinkEncPad` [0.. 3] .

The `MIC` is calculated over all the fields in the frame.

msg = `DevAddr` | `WorUplinkEnc` | `WFCnt`

The MIC SHALL be calculated as follows [RFC4493]:

`CMAC` = aes128_cmac(`WorSIntKey`, B_0 | *msg*)
`MIC` = `CMAC`[0..3] ,

where block B_0 is defined as follows:

Size (octets)	1	4	1	4	4	1	1
B0	0x49	4x 0x00	0x00 (Dir)	DevAddr	WFCnt32	0x00	0x0E (len(WOR Frame))

Table 11: B0 block for MIC computation in WOR Relay Class A Uplink

where len(*WOR Frame*) denotes the length of the frame in octets.

6 WOR ACK Detail

6.1 Physical Parameters

Refer to [RP002] for more information on the physical parameters.

6.2 Payload Format

The WOR ACK payload depends on the type of WOR frame being acknowledged. Table 12 shows the format of WOR ACK in response to a WOR Relay Class A Uplink.

Size (octets)	3	4
WOR ACK Uplink	AckUplinkEnc	MIC

Table 12: WOR ACK Relay Class A Uplink format

The AckUplinkEnc field provides information about the configuration of the relay. This field is the result of encrypting the AckUplink field with the WorSEncKey. This field conveys the internal state of the relay.

Size (octets)	3
AckUplink	StateSync

Table 13: AckUplink format

Where:

Bits	23:22	21:20	19:16	15:14	13:11	10:0
StateSync	CadToRx	Forward	RelayDataRate	XTALAccuracy	CADPeriodicity	TOffset

Table 14:StateSync format

The CadToRx field describes the delay between the start of CAD and the start of the reception process.

CadToRx	Description
0	2 symbols
1	4 symbols
2	6 symbols
3	8 symbols

Table 15: CadToRx decoding

The end-device MAY use this information to optimize the preamble length of the WOR frame.

The Forward field indicates the current state of the relay forward limitations.

Forward	Description
0	Forward limit is OK.
1	Forward limit is reached. Retry in 30 min.
2	Forward limit is reached. Retry in 60 min.
3	Forward is disabled.

Table 16: Forward decoding

The RelayDataRate field defines the data rate used by the relay to forward the end-device payload. Refer to [RP002] for more information on the bandwidth and spreading factor. This field SHALL be used by the end-device to limit the payload size. Refer to Section 8.3, “Uplink Maximum Payload Size” for more information.

`XTALAccuracy` encodes the crystal frequency accuracy of the relay clock. It MAY be used by the end-device to compute the preamble length and the receive windows duration.

XTALAccuracy	**Description**
0	Better than 10 ppm
1	Better than 20 ppm
2	Better than 30 ppm
3	Better than 40 ppm

Table 17: XTALAccuracy decoding

`CADPeriodicity` encodes the delay between two consecutive CAD scans performed by the relay on a single channel.

CADPeriodicity	**Description**
0	1 second
1	500 milliseconds
2	250 milliseconds
3	100 milliseconds
4	50 milliseconds
5	20 milliseconds
6	RFU
7	RFU

Table 18: CADPeriodicity decoding

`TOffset` encodes the time, in milliseconds, between the start of the scan and the end of the reception of the WOR frame preamble. This is illustrated in the following diagram:

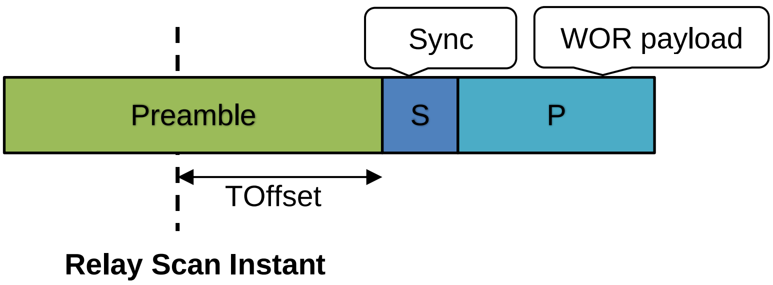

Figure 14: TOffset computation

`AckUplink` is encrypted.as shown in Table 19; for each ACK frame, the algorithm defines a block A_{ACK}:

Size (octets)	1	2	1	4	4	3	1
A_{ACK}	0x01	2x 0x00	1 (Downlink)	DevAddr	WFcnt32	WOR ACK Frequency	WOR ACK Datarate

Table 19: Block A_{ACK} format

The block A_{ACK} SHALL be encrypted to obtain a block S_{ACK}.

S_{ACK} = aes128_encrypt(`WorSEncKey`, A_{ACK})

Encryption and decryption of the payload SHALL be calculated as follows:

`AckUplinkEncPad` = (`AckUplink` | pad16) xor S_{ACK}
`AckUplinkEnc` = `AckUplinkEncPad` [0.. 2] .

The `MIC` is calculated over all the fields in the frame and SHALL be calculated as follows [RFC4493]:

`CMAC` = aes128_cmac(`WorSIntKey`, B_0 | `AckUplinkEnc` | *WOR* | pad_{16})
`MIC` = `CMAC`[0..3]

Where:

- WOR contains the data received in the WOR Relay Class A Uplink and is defined as:

Size (bits)	4	4	24	16	32
WOR	`RFU`	`Datarate`	`Frequency`	`WFCnt`	`DevAddr`

Table 20: WOR block for MIC computation in WOR ACK

- Block B_0 is defined as follows:

Size (octets)	1	4	1	4	4	1	1
B0	`0x49`	4x `0x00`	`0x01` (Dir)	`DevAddr`	`WFCnt32`	`0x00`	`0x07` (len(ACK))

Table 21: B0 block for MIC computation in WOR ACK

`DevAddr` is the 32-bit device address of the end-device as defined in [TS001].

`WFCnt32` is the 32-bit counter used to compute the `MIC` in the received WOR frame.

7 RXR Window

7.1 Physical Parameters

Refer to [RP002] for more information on the physical parameter.

Like RX1 and RX2, the RXR window start time is defined using the end of the end-device's LoRaWAN uplink transmission as a reference.

The RXR (RX Relay) window (if opened) SHALL be opened no later than the RXR_DELAY, after the end of the uplink modulation.

> **Note:** In LoRaWAN 1.1.X or higher, if a Network Server intends to have a downlink transmitted to an end-device on RXR, it SHALL use the RXR physical parameters to compute the MIC of that downlink.

7.2 Window Opening Condition

The end-device SHOULD open the RXR slot after a WOR exchange, when no valid downlink was received in the RX1 or RX2 slots.

The duration of the receive window SHALL be at least the time required by the end-device's radio transceiver to detect a downlink preamble starting at the RXR_DELAY after the end of the uplink modulation.

If a preamble is detected during RXR windows, the end-device's radio transceiver SHOULD remain active until the downlink frame is demodulated.

The end-device SHALL stop to demodulate the incoming message if it is too long regarding the maximum payload size authorized at this data rate.

As specified in [TS001], the end-device receive window duration needs to accommodate for the maximum potential inaccuracy of the end-device's clock. In case of the RXR windows, the end-device SHALL also take into account the potential inaccuracy of the relay's clock. Refer to `StateSync` received in the WOR ACK frame.

8 Message Forwarding

The followings sections present the encapsulation mechanisms and forwarding limitations. The payload construction is defined in section 9.

> **Note:** Although this document only describes encapsulation-based forwarding, other mechanisms (such as direct physical forwarding) are possible, assuming that the Network Server is aware of them. Those mechanisms are not specified by the LoRa Alliance.

8.1 Uplink Encapsulation

As the relay is a LoRaWAN Class A end-device, it SHALL use its own LoRaWAN session key to forward the end-device uplink and the associated metadata to the Network Server.

The end-device payload that will be forwarded to the Network Server SHALL be the `PHYPayload` as defined in [TS001]. The `PHYPayload` is composed of the `MHDR`, `MACPayload`, and `MIC`.

The following metadata will be forwarded to the Network Server:

- Received Signal Strength Indication (RSSI) of the LoRaWAN uplink
- Signal-to-Noise Ratio (SNR) of the LoRaWAN uplink
- RX frequency of the LoRaWAN uplink
- RX data rate of the LoRaWAN uplink
- WOR channel used for the WOR frame

Byte order and encoding are defined in Section 9.1.

8.2 Uplink Forwarding Timing

Once the relay has received an uplink from an end-device, and if authorized to forward it, it SHALL forward it to the Network Server after the RELAY_FWD_DELAY, with the end of the uplink reception being the time reference.

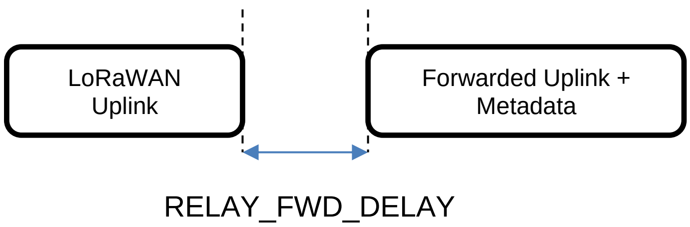

Table 22: Forwarded uplink timing

8.3 Uplink Maximum Payload Size

The relay uplink forwarding capability is driven by the data rate between itself and the Network Server. The end-device SHALL adapt its payload size to match the forwarding capability of a relay taking into account the forwarded metadata overhead (See 9.1)

> **Note:** It is up to the network operator to ensure that the data rate between the relay and the Network Server is good enough to allow proper operation.

The relay notifies the end-device of its forwarding capability in the WOR ACK frame with the field `RelayDataRate`.

If a relay receives a frame that it is not able to forward, it SHALL drop it.

Refer to [RP002] for the maximum payload size allowed for each data rate for an end-device under a relay. This information is defined for each region in the section "Maximum payload size" and is noted "repeater compatible".

8.4 Downlink Encapsulation

If the Network Server intends to have a downlink sent to the end-device by the relay, it SHALL send to the relay the `PHYPayload` to be forwarded downlink.

> **Note**: As stated in Section 3.5, it needs to do so on the RX1 or RX2 windows of the forwarded uplink.

The `PHYPayload` is defined in [TS001]. It is composed of the `MHDR`, `MACPayload`, and `MIC`.

Byte order and encoding are defined in Section 9.2.

If the relay decides to forward the downlink to the end-device, it needs to use the RXR window as stated in Section 3.6.

8.5 Downlink Maximum Payload Size

When building a downlink for an end-device, the Network Server SHALL take into consideration the data rate between the network and the relay but also between the relay and the end-device. The maximum payload size will be the smallest value generated by these two data rates.

Refer to [RP002] for the maximum payload size allowed for each data rate. The maximum payload size that can be forwarded to an end-device is defined for each region in the section "Maximum payload size" and is noted "repeater compatible".

8.6 Forward/Filter Join-Request

A relay SHALL be able to filter Join-Requests based on their `JoinEUI` and `DevEUI`.

The forward/filter algorithm used is the longest prefix match (LPM).

This lookup table is filled with:

- Concatenation of `JoinEUI` and `DevEUI`
- Rules length (0 to 16)
- Rules (forward or filter)

By default, if no rule matches the Join-Request, the relay SHOULD forward it.

The relay SHALL support a lookup table of 16 inputs.

> **Note:** A relay can also filter all the Join-Requests that do not match the lookup table by updating rule 0 to “filter". Refer to 10.3 “Manage Join-Request Forward List (*FilterListReq*, *FilterListAns*)” for more information.

Refer to “APPENDIX 3: Filter/Forward Example” for an example.

8.7 Notify Standard Class A Uplink

If a relay is not able to check the MIC validity of the WOR Relay Class A Uplink frame, it SHALL notify the Network Server that it has detected a new end-device, if the forwarding limitation has not been reached.

8.8 Forwarding Limitations

A relay SHALL implement a function limiting the uplink forwarding. This function is based on a token bucket system; a relay SHALL only forward the uplink when it has at least 1 token remaining for that type of message, and it SHALL decrement the associated bucket by 1 token while doing so.

The token bucket algorithm relies on two parameters:
- Reload rate (X tokens per hour)
- Bucket size (number of tokens that a bucket can hold)

X tokens SHALL be added to the bucket every hour. If a token arrives when the bucket is full (i.e., the token counter is equal to the bucket size), it SHALL be discarded.

If the bucket is empty, the relay SHALL NOT listen to the end-device uplink.

For each type of message, the token bucket algorithm parameters MAY have different values. Table 23 shows the different message types and the default values for each parameter.

Message Type	Reload Rate (X per Hour)	Bucket Size
Join-Request	4	8
New device detected	4	8
Class A Uplink (per trusted end-device)	No default value	
Class A Uplink (for all trusted end-devices)	8	16
Global uplink	8	16

Table 23: Forwarding limit list

When the relay starts the channel scan, every bucket counter SHALL be set to the default reload rate value.

Some messages consume tokens from different buckets at the same time. Table 24 shows how each bucket is impacted by each type of message.

		Limit Type				
		Join-Request	Class A Uplink (per trusted end-device)	New device detected	Class A Uplink (for all trusted end-device)	Global uplink
Message Type	Join-Request	x				x
	Class A Uplink from trusted end-device		x		x	x
	Class A Uplink from unknown end-device			x		x

Table 24: Message limits

> **Note 1**: A trusted end-device is notified of the forwarding limitation status with the field `Forward` in the WOR ACK frame.
>
> **Note 2:** The duty cycle limitation supersedes these limitations.
>
> **Note 3:** The Network Server can update these limitations during the lifetime of a relay based on its knowledge of the end-device's deployment.

The reload rate parameter can be set to "no limit". In this case, the bucket size parameter SHALL be ignored and no limit SHALL be applied for this type of message.

Refer to Section 10.4, "Add End-Device to Uplink Forward List (*UpdateUplinkListReq*, *UpdateUplinkListAns*)" and Section 10.6, "Manage Forwarding Limitations (*ConfigureFwdLimitReq*, *ConfigureFwdLimitAns*)" for more information on how to update the limits value.

9 Forwarded Message Protocol

All the traffic forwarded by/to a relay SHALL use LA_FPORT_RELAY (see [TS008] *FPort Assignments*) for the exchange between the relay and the Network Server. This port SHALL NOT be used for any other purposes.

A frame using LA_FPORT_RELAY SHALL NOT encapsulate more than one forwarded LoRaWAN frame.

All messages exchanged on LA_FPORT_RELAY are encrypted using the network session key of the relay. Table 25 describes the key to use, depending on the LoRaWAN version.

LoRaWAN Version	Description
1.0.X	NwkSKey is used to encrypt and compute the MIC of forwarded messages.
1.1.X or higher	F/SNwkSIntKey is used to compute the MIC of the frame. NwkSEncKey is used to encrypt the forwarded message.

Table 25: Network session key vs LoRaWAN version

9.1 Forward Uplink from an End-Device (*ForwardUplinkReq*)

The relay SHALL forward a LoRaWAN uplink by sending a LoRaWAN uplink frame to the network with *FPort* equal to LA_FPORT_RELAY and the payload described in Table 26.

Size (octets)	3	3	variable
***ForwardUplinkReq* payload**	`Uplink Metadata`	`Uplink Frequency`	`Uplink Payload`

Table 26: ForwardUplinkReq format

Where:

`Bits`	`23:18`	`17:16`	`15:9`	`8:4`	`3:0`
`UplinkMetadata`	`RFU`	`WOR Channel`	`Uplink RSSI`	`Uplink SNR`	`Uplink Datarate`

Table 27: UplinkMetadata format

The `WORChannel` field encodes the WOR channel that has been used for the WOR communication:

`WORChannel`	Description
`0`	Default channel has been used
`1`	Second channel has been used
`2..3`	RFU

Table 28: WORChannel decoding

The `UplinkRSSI` SHALL encode the RSSI of the LoRaWAN uplink received by the relay according to the following:

$$\text{UplinkRSSI}_{\text{dBm}} = -1 \times \text{UplinkRSSI} - 15;$$

The relay may receive an uplink with an RSSI greater than -15dBm or lower than -142dBm. In this case, the relay SHALL set the `UplinkRSSI` to the closest possible value.

The `UplinkSNR` field SHALL encode the SNR of the LoRaWAN uplink received by the relay according the following formula:

$$\text{UplinkSNR}_{\text{dB}} = \texttt{UplinkSNR} - 20$$

The relay may receive an uplink with an SNR greater than 11dB or lower than -20dB. In this case, the relay SHALL set the `UplinkSNR` to the closest possible value.

The `UplinkDatarate` indicates the data rate used by the LoRaWAN uplink received by the relay. Refer to [RP002] for more information on the bandwidth and spreading factor.

The `UplinkFrequency` field encodes the frequency used for the LoRaWAN uplink received by the relay. The frequency SHALL be encoded following the convention defined in the [TS001] ***NewChannelReq*** command.

The `UplinkPayload` field is the `PHYPayload`, as defined in [TS001] of the LoRaWAN uplink received by the relay.

9.2 Downlink to a Relay for Further Forwarding (*ForwardDownlinkReq*)

A relay receiving a downlink frame with *FPort* equal to LA_FPORT_RELAY SHALL attempt to forward its content downlink, after sanity checking.

Size(bit)	variable
Field	`Downlink Payload`

Table 29: ForwardDownlinkReq

The `DownlinkPayload` field is the `PHYPayload`, as defined in [TS001], to be forwarded by the relay to the end-device in the RXR window.

The output power of the forwarded downlink SHALL be the default one.

10 Relay MAC Commands

The commands to configure the relay-related parameters on the end-device and the parameters of the relay itself are MAC commands. Refer to [TS001] for more information on how to build a MAC command.

Table 30 summarizes the list of relay MAC commands:

CID	Relay Protocol Command Name	From: End-device	From: Relay	From: Network Server	Short Description
0x40	***RelayConfReq***			x	Configure the relay radio parameter of the relay
0x40	***RelayConfAns***		x		Conveys the answer to ***RelayConfReq***
0x41	***EndDeviceConfReq***			x	Configure the relay parameter of the end-device
0x41	***EndDeviceConfAns***	x			Conveys the answer to ***EndDeviceConfReq***
0x42	***FilterListReq***			x	Update the list of forwarding/filter Join-Requests
0x42	***FilterListAns***		x		Conveys the answer to ***FilterListReq***
0x43	***UpdateUplinkListReq***			x	Add an end-device to the trusted end-device list
0x43	***UpdateUplinkListAns***		x		Conveys the answer to ***UpdateUplinkListReq***
0x44	***CtrlUplinkListReq***			x	Remove an end-device from the trusted end-device list
0x44	***CtrlUplinkListAns***		x		Conveys the answer to ***CtrlUplinkListReq***
0x45	***ConfigureFwdLimitReq***			x	Configure forwarding limitation
0x45	***ConfigureFwdLimitAns***		x		Conveys the answer to ***ConfigureFwdLimitReq***
0x46	***NotifyNewEndDeviceReq***		x		Notify the network that a new end-device appeared under a relay.

Table 30: Summary of relay protocol command messages

10.1 Update Relay Configuration (*RelayConfReq, RelayConfAns*)

The ***RelayConfReq*** command is sent by the Network Server to the relay.

The ***RelayConfReq*** command updates the relay radio configuration of the relay. If the relay is already started, the relay SHALL apply the new configuration on the next scan slot (up to 1 second later).

Size (octets)	2	3
***RelayConfReq* payload**	`ChannelSettingsRelay`	`SecondChFreq`

Table 31: RelayConfReq format

Where:

Bits	15:14	13	12:10	9	8:7	6:3	2:0
Channel SettingsRelay	RFU	StartStop	CADPeriodicity	DefaultChIdx	SecondChIdx	SecondChDr	SecondChAckOffset

Table 32: ChannelSettingsRelay format

The `StartStop` field indicates to the relay whether to enable or disable its relay activity. If this field is set to 0, all the other fields in the ***RelayConfReq*** command SHALL be ignored.

StartStop	Description
0	Disable the relay
1	Enable the relay

Table 33: StartStop decoding

The `CADPeriodicity` field indicates the delay between two scans on the default channel. This field uses the same encoding as the WOR ACK frame, refer to "Table 18: CADPeriodicity decoding" for the possible values.

The `DefaultChIdx` field indicates the channel used by the default channel. Refer to [RP002] for more information.

The `SecondChIdx` field indicates if a second channel is used. This field is encoded as follows:

SecondChIdx	Description
0	No second channel is used
1	Second channel settings
2..3	RFU

Table 34: Second channel decoding

The `SecondChDr` field indicates the data rate used for the second channel. Refer to [RP002] for more information on the bandwidth and spreading factor.

The `SecondChAckOffset` field indicates the frequency offset used for the second channel WOR ACK frame. This field is encoded as follows:

SecondChAckOffset	Description
0	0 kHz
1	200 kHz
2	400 kHz
3	800 kHz
4	1600 kHz
5	3200 kHz
6..7	RFU

Table 35: WOR ACK frequency offset

The frequency used for the WOR ACK frame SHALL be encoded using the following formula:

$$\text{WOR ACK}_{FREQUENCY\_Hz} = \texttt{SecondChFreq}_{Hz} + \texttt{SecondChAckOffset}_{Hz}$$

If `SecondChIdx` is set to 0, `SecondChDr`, `SecondChACKOffset`, and `SecondChFreq` SHALL be ignored.

The `SecondChFreq` field indicates the frequency used for the second channel. The frequency is encoded following the convention defined in the [TS001] ***NewChannelReq*** command.

The relay SHALL respond to this command with the ***RelayConfAns*** command with the following payload:

Size (octets)	1
***RelayConfAns* payload**	Status

Table 36: RelayConfAns format

Where:

Bits	7:6	5	4	3	2	1	0
Status	RFU	CADPeriodicityACK	DefaultChIdxACK	SecondChIdxACK	SecondChDrACK	SecondChAckOffsetACK	SecondChFreqACK

Table 37: Status format

The ***RelayConfAns*** `Status` bits have the following meaning:

- Bit = 0 means the parameter has an invalid value.
- Bit = 1 means the parameter has a valid value.

If any of the bits equal `0`, the command was not successful, and the relay SHALL discard the whole command.

10.2 Update End-Device Configuration (*EndDeviceConfReq*, *EndDeviceConfAns*)

The ***EndDeviceConfReq*** command is sent by the Network Server to the end-device.

The ***EndDeviceConfReq*** command updates the relay configuration on an end-device.

Size (octets)	1	2	3
***EndDeviceConfReq* payload**	ActivationRelayMode	ChannelSettingsED	SecondChFreq

Table 38: EndDeviceConfReq format

Where:

Bits	7:4	3:2	1:0
ActivationRelayMode	RFU	RelayModeActivation	SmartEnableLevel

Table 39: ActivationRelayMode format

The `RelayModeActivation` field indicates how the end-device SHOULD manage the relay mode.

RelayModeActivation	**Description**
0	Disable the relay mode
1	Enable the relay mode
2	Dynamic
3	End-device controlled

Table 40: RelayModeActivation format

If `RelayModeActivation` is set to 3, the end-device is free to enable, or not, the relay mode on its own. This is the default mode.

If `RelayModeActivation` is set to 0, all of the other fields in the command ***EndDeviceConfReq*** SHALL be ignored and the end-device SHALL disable the relay mode.

If `RelayModeActivation` is set to 1, the end-device SHALL enable the relay mode even if the end-device does not receive a WOR ACK or a downlink on RXR.

If `RelayModeActivation` is set to 2, the end-device SHALL be able to automatically enable the relay mode if the end-device does not receive a downlink after several uplinks.

If `RelayModeActivation` is set to 2, the `SmartEnableLevel` field indicates that the relay mode SHALL be enabled if the end-device does not receive a valid downlink after *X* consecutive uplinks.

SmartEnableLevel	**Description**
0	8
1	16
2	32
3	64

Table 41: SmartEnableLevel format

The `RelayActivation` field is valid only for the current LoRaWAN session, i.e., if the end-device sends a new Join-Request, the end-device SHALL return to the “end-device controlled” mode.

Bits	15	14:9	8:7	6:3	2:0
ChannelSettingsED	RFU	BackOff	SecondCh Idx	SecondCh Dr	SecondCh AckOffset

Table 42: ChannelSettingsED format

The fields `SecondChIdx`, `SecondChDr`, `SecondChAckOffset`, and `SecondChFreq` have the same meaning as those described in section “10.1 Update Relay Configuration (*RelayConfReq, RelayConfAns*)”.

The `BackOff` field indicates how the end-device SHALL behave when it does not receive a WOR ACK frame.

BackOff	**Description**
0	Always send a LoRaWAN uplink
1..63	Send a LoRaWAN uplink after *X* WOR frames without a WOR ACK

Table 43: BackOff decoding

The end-device SHALL respond to this command with the ***EndDeviceConfAns*** command with the following payload:

Size (octets)	1
***EndDeviceConfAns* payload**	Status

Table 44: EndDeviceConfAns format

Where:

Bits	7:4	3	2	1	0
Status	RFU	BackOff ACK	SecondCh IdxACK	SecondCh DrACK	SecondCh FreqACK

Table 45: Status format

The ***RelayConfAns*** `Status` bits have the following meaning:

- Bit = 0 means the parameter has an invalid value.
- Bit = 1 means the parameter has a valid value.

If any of the bits equal `0`, the command was not successful, and the end-device SHALL discard the whole command.

10.3 Manage Join-Request Forward List (*FilterListReq*, *FilterListAns*)

The ***FilterListReq*** command is sent by the Network Server to the relay.

The ***FilterListReq*** command updates the filter and forward list for the Join-Request as described in Section 8.6 “Forward/Filter .

Size (octets)	2	N
***FilterListReq* payload**	FilterList Param	FilterList Eui

Table 46: FilterListReq format

Where:

Bits	15:11	10:7	6:5	4:0
FilterListParam	RFU	FilterList Idx	FilterList Action	FilterList Len

Table 47: FilterListParam format

The `FilterListIdx` field indicates the rules index that will be updated.

The `FilterListAction` field indicates the action associated to the Extended Unique Identifier (EUI).

FilterListAction	Description
0	No Rule
1	Forward
2	Filter
3	RFU

Table 48: FilterListAction decoding

The `FilterListLen` field indicates the length, *N*, of the field `FilterListEui`, $0 \leq N \leq 16$.

The `FilterListEui` field is an *N*-byte field representing the leftmost *N* bytes of the concatenation of `JoinEUI` and `DevEUI`.

An error SHALL be generated if:

- `FilterListLen` is equal to 17 or more.
- `FilterListLen` is equal to 0 and `FilterListIdx` is different from 0.

- `FilterListIdx` is equal to 0 and `FilterListLen` is different from 0 and `FilterListAction` is different from 1 or 2.
- `FilterListAction` is equal to “No Rule” and `FilterListLen` is different from 0.

The relay SHALL respond to this command with the ***FilterListAns*** command with the following payload:

Size (octets)	1
***RelayConfAns* payload**	Status

Table 49: *FilterListAns* format

Where:

Bits	7:5	2	1	0
Status	RFU	CombinedRulesACK	FilterListLenACK	FilterListActionACK

Table 50: Status format

The ***RelayConfAns*** `Status` bits have the following meaning:

- Bit = 0 means the parameter has an invalid value.
- Bit = 1 means the parameter has a valid value.

The `CombinedRulesACK` field indicates if all the parameters combined create a valid rule or not.

If any of the bits equal `0`, the command was not successful and the Relay SHALL discard the whole command.

10.4 Add End-Device to Uplink Forward List (*UpdateUplinkListReq*, *UpdateUplinkListAns*)

The ***UpdateUplinkListReq*** command is sent by the Network Server to the relay.

The ***UpdateUplinkListReq*** command updates the internal list of trusted end-devices.

> **Note:** A Network Server has to be careful not to enable an end-device on multiple relays that can interfere with one another, as it could result in a collision of the WOR ACKs.

Size (octets)	1	1	4	4	16
***UpdateUplinkListReq* payload**	UplinkListIdxPL	UplinkLimitPL	DevAddr	WFCnt32	RootWorSKey

Table 51: UpdateUplinkListReq format

Where:

Bits	7:4	3:0
UplinkListIdxPL	RFU	UplinkListIdx

Table 52: UplinkListIdxPL format

The `UplinkListIdx` field indicates which rule will be updated.

Bits	7:6	5:0
UplinkLimitPL	UplinkLimit BucketSize	UplinkLimit ReloadRate

Table 53: UplinkLimitPL format

The `UplinkLimitReloadRate` field indicates how many tokens this end-device will earn every hour.

UplinkLimit ReloadRate	Description
0 to 62	*X* tokens every hour
63	No limitation (forward all valid uplinks)

Table 54: UplinkLimitReloadRate decoding

The `UplinkLimitBucketSize` field indicates the multiplier to determine the bucket size according to the following formula:

BucketSize $_{\text{TOKEN}}$ = `UplinkLimitReloadRate x UplinkLimitBucketSize`

UplinkLimit BucketSize	Description
0	1
1	2
2	4
3	12

Table 55: UplinkLimitBucketSize decoding

The `DevAddr` consists of 32 bits and identifies the end-device within the current network. Refer to [TS001- Section 6.1.1] for further details.

The `WFCnt32` field is a 32-bit unsigned integer and represents the last known value by the Network Server of the WOR frame counter. This field is used to compute the MIC in the WOR Relay Class A Uplink and WOR ACK frame.

The `RootWorSKey` is the key associated to the `DevAddr`. Refer to Section 4 for more information.

When this command is received, the credit counter SHALL be set to `UplinkLimitReloadRate x UplinkLimitBucketSize.`

Overwriting an existing index is allowed; no error SHALL be generated.

The relay SHALL answer this command with the ***UpdateUplinkListAns*** command. The ***UpdateUplinkListAns*** MAC reply contains no payload.

10.5 Manage End-Device from Uplink Forward List (*CtrlUplinkListReq*, *CtrlUplinkListAns*)

The ***CtrlUplinkListReq*** command is sent by the Network Server to the relay.

The ***CtrlUplinkListReq*** command updates the internal list of trusted end-devices. It allows the Network Server to remove (or not) an end-device or get the last valid `WFcnt32`.

Size (octets)	1
***CtrlUplinkListReq* payload**	`CtrlUplinkActionPL`

Table 56: CtrlUplinkListReq format

Where:

Bits	7:5	4	3:0
CtrlUplinkActionPL	RFU	CtrlUplinkAction	UplinkListIdx

Table 57: CtrlUplinkActionPL format

The `CtrlUplinkAction` field requests the relay to read its `WFCnt32` related to an end-device or to remove the end-device from the trusted list.

CtrlUplinkAction	Description
0	Only read end-device `WFCnt32`
1	Remove the end-device from the trusted list

Table 58: CtrlUplinkAction decoding

The `UplinkListIdx` field indicates which entry to operate on.

An error SHALL be generated if this command is applied to an empty entry and the relay SHALL discard the command.

The relay SHALL answer this command with the ***CtrlUplinkListAns*** command with the following payload:

Size (octets)	1	4
***CtrlUplinkListAns* payload**	Status	WFCnt32

Table 59: CtrlUplinkListAns format

Where:

Bits	7:1	0
Status	RFU	UplinkListIdxACK

Table 60: Status format

The ***CtrlUplinkListAns*** `Status` bits have the following meaning:

- Bit = 0 means the parameter has an invalid value.
- Bit = 1 means the parameter has a valid value.

The `UplinkListIdxACK` field indicates if a valid index (i.e., an index to a non-empty entry) has been selected.

If any of the bits equal `0`, the command was not successful.

The `WFCnt32` field is a 32-bit unsigned integer and represents the last valid value received by the relay. This field SHALL be ignored if an error is generated.

10.6 Manage Forwarding Limitations (*ConfigureFwdLimitReq*, *ConfigureFwdLimitAns*)

The ***ConfigureFwdLimitReq*** command is sent by the Network Server to the relay.

On receiving a ***ConfigureFwdLimitReq*** command, a relay SHALL update its forwarding limitation function as defined in Section 8.8 “Forwarding Limitations”.

Size (octets)	4	1
***ConfigureFwdLimitReq* payload**	FwdLimit ReloadRatePL	FwdLimit LoadCapacityPL

Table 61: ConfigureFwdLimitReq format

Where:

Bits	31:30	29:28	27:21	20:14	13:7	6:0
FwdLimit ReloadRatePL	RFU	ResetLimit Counter	JoinReq Reload Rate	Notify Reload Rate	GlobalUplink ReloadRate	Overall Reload Rate

Table 62: `FwdLimitReloadRatePL` format

The `ResetLimitCounter` field instructs the relay to reset (or not) all the token counters when it receives this command.

ResetLimitCounter	**Description**
0	Set the token counter to 0
1	Set the token counter to ReloadRate
2	Set the token counter to the maximum value
3	Do not change the token counter

Table 63: ResetLimitCounter decoding

The `JoinReqReloadRate`, `NotifyReloadRate`, `GlobalUplinkReloadRate`, and `OveralReloadRate` indicate how many tokens will be added every hour for the various types of messages. They are encoded per Table 64:

ReloadRate	**Description**
0 to 126	*X* credits per hour
127	No limitation (forward all valid messages)

Table 64: Forward Limit decoding

The `JoinReqReloadRate` field is the limit for Join-Request messages forwarded to the network.

The `NotifyReloadRate` field is the limit for unknown end-device notifications. Refer to Section 8.7, "Notify Standard Class A Uplink" and Section 10.7, "Notify New End-Device Under a Relay (*NotifyNewEndDeviceReq*)" for more information.

The `GlobalUplinkReloadRate` field is the limit for uplinks forwarded to the network for all trusted end-devices.

The `OverallReloadRate` field is the global limit for a relay. This limit includes all of the following messages: Join-Request, uplink from trusted end-device, and notification.

Bits	7:6	5:4	3:2	1:0
FwdLimit LoadCapacity PL	JoinReq LimitSize	Notify LimitSize	GlobalUplink LimitSize	Overall LimitSize

Table 65: FwdLimitLoadCapacityPL format

The fields in `FwdLimitLoadCapacityPL` indicate the duration during which the different limits can accumulate credit. Refer to “Table 55: UplinkLimitBucketSize decoding” to decode these fields.

The relay SHALL answer this command with the ***ConfigureFwdLimitAns*** command. The ***UpdateUplinkListAns*** MAC reply contains no payload.

10.7 Notify New End-Device Under a Relay (*NotifyNewEndDeviceReq*)

The ***NotifyNewEndDeviceReq*** command is sent by the relay to the Network Server.

The ***NotifyNewEndDeviceReq*** command is used by the relay when it receives a WOR Relay Class A Uplink for which it is not able to verify the MIC.

Size (octets)	4	2
***NotifyNewEndDeviceReq* payload**	`DevAddr`	`PowerLevel`

Table 66: NotifyNewEndDeviceReq format

The `DevAddr` consists of 32 bits advertised by the end-device in the WOR frame for which the relay has been unable to verify the MIC.

Bits	15:12	11:5	4:0
PowerLevel	RFU	WORRSSI	WORSNR

Table 67: PowerLevel format

The `WORRSSI` SHALL encode the RSSI of the WOR frame received by the relay according to the following:

$\text{WOR RSSI}_{\text{dBm}} = -1 \times \text{WORRSSI} - 15$

The relay may receive a WOR frame with an RSSI greater than -15 dBm or lower than -142 dBm. In this case, the relay SHALL set the `WORRSSI` to the closest possible value.

The `WORSNR` field SHALL encode the SNR of the WOR frame received by the relay according the following formula:

$\text{WOR SNR}_{\text{dB}} = \text{WORSNR} - 20$

The relay may receive an uplink with an SNR greater than 11 dB or lower than -20 dB. In this case, the relay SHALL set the `WORSNR` to the closest possible value.

11 Certification

11.1 Relay Certification

The relay SHALL be certified as a LoRaWAN Class A end-device.

The relay MAY be certified as a LoRaWAN Class B end-device.

The relay MAY be certified as a LoRaWAN Class C end-device.

The certification of a relay as a relay will be defined in a different document.

11.2 End-Device Certification

The certification of an end-device under a relay will be defined in a different document.

12 Glossary

ABP	Activation By Personalization
AES	Advanced Encryption Standard
AS	Application Server
CAD	Channel Activity Detection
DoS	Denial-of-Service Attack
EUI	Extended Unique Identifier
FSK	Frequency-Shift Keying modulation technique
JS	Join Server
KEK	Key-Encryption-Key
LPM	Longest Prefix Match
LBT	Listen Before Talk
LoRa	Long Range modulation technique
LoRaWAN	Long Range Network Protocol
LR-FHSS	Long Range-Frequency Hopping Spread Spectrum
MAC	Medium Access Control
MIC	Message Integrity Code
NS	Network Server
OTAA	Over-The-Air Activation
ppm	Parts per million
RFU	Reserved for Future Usage
RSSI	Received Signal Strength Indication
RX	Receiver
RXR	RX Relay
SNR	Signal-to-Noise Ratio
WOR	Wake On Radio
WOR ACK	Wake On Radio Acknowledge

13 Bibliography

13.1 References

[RP002]: RP002-1.0.2 LoRaWAN Regional Parameters, LoRa Alliance Technical Committee, October 2020

[TS001] LoRaWAN® MAC Layer Specification, v1.0 through V1.1, the LoRa Alliance.

[TS008] LoRa Alliance Assigned Value Registries, May 2020

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997

[RFC4493] Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The AES-CMAC Algorithm", RFC 4493, DOI 10.17487/RFC4493, June 2006

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017

APPENDIX 1: RELAY/END-DEVICE SYNCHRONIZATION

The timing synchronization mechanism only applies to synchronized devices as defined in Section 3.9.

In order to minimize the WOR frame preamble length, a synchronized end-device has to estimate when the relay scans the default and second channels. To do that it needs to:

- Use the information conveyed by the ACK frame (`CADPeriodicity`, `TOffset`, …)
- Estimate the drift due to crystal accuracy

The end-device has to send a WOR frame with a preamble long enough for the relay to listen to at least to 6 + `CadToRx` symbols.

> **Note:** 6 symbols are the minimum required by the relay radio transceiver to demodulate an incoming message and `CadToRx` is the number of symbols needed by the relay to switch from CAD to RX mode.

Figure 15 illustrates the median and worst synchronization cases:

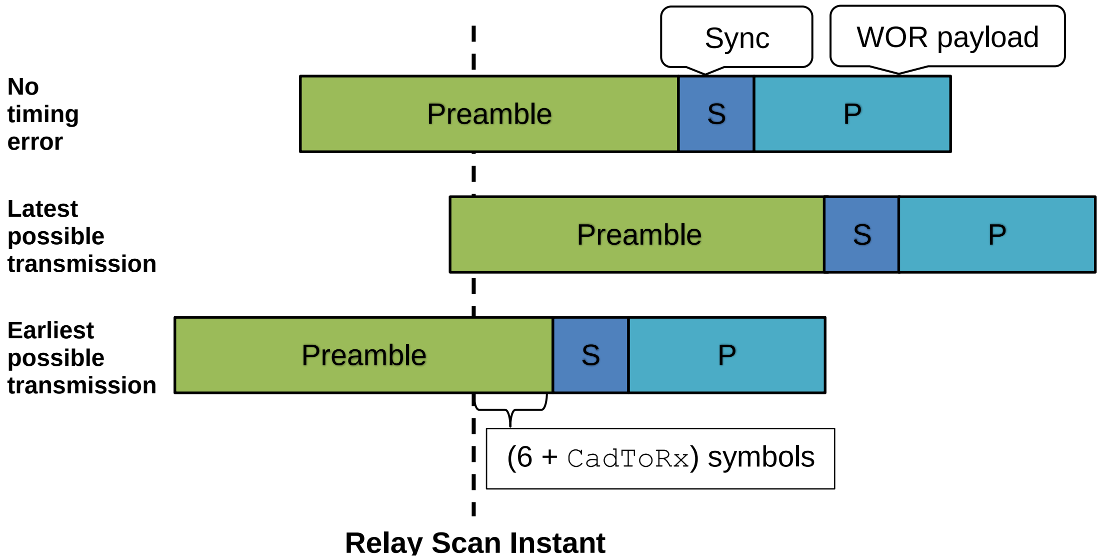

Figure 15: Synchronization median and worst cases

In the “earliest” case, the end-device clock ticks faster than the relay clock, therefore the source device sends ahead of schedule.

In the “latest” case, the end-device clock ticks slower than the relay clock, therefore the source device sends late compared to the ideal time.

The minimum preamble length of a WOR frame consists of 8 symbols.

Initially, the end-device is in the **initialized** state and the preamble length is set to the maximum preamble length.

An end-device in the **synchronized** state has to estimate the next wake-up frame transmission opportunity using the following formula:

$$\boldsymbol{T_{NEXT}} = N * CADPeriodicity + T_{REF}$$

With *N* the slot index:

$$\boldsymbol{N} = ceil\left(\frac{T_{NOW} - T_{REF}}{CADPeriodicity}\right)$$

$$\boldsymbol{T_{REF}} : Last\ valid\ relay\ scan\ time(formula\ at\ the\ end\ of\ this\ appendix)$$

Then, the end-device has to estimate the maximum drift error and the start time of the WOR payload.

$$\boldsymbol{DriftError} = \frac{(RelayXtalAccuracy + EndDeviceXtalAccuracy) * (T_{NEXT} - T_{REF})}{1000000}$$

with *RelayXtalAccuracy* and *EndDeviceXtalAccuracy* expressed in ppm and *DriftError* in milliseconds.

$$\boldsymbol{T_{START}} = T_{NEXT} - \frac{DriftError}{2}$$

> **Note 1:** If the *DriftError* is greater than `CADPeriodicity`, the end-device goes into the unsynchronized state.
>
> **Note 2:** If T_{NOW} is greater than T_{START}, increment *N* by 1 and re-compute the previous formula.
>
> **Note 3:** If T_{START} is too close to T_{NOW}, the end-device can decide to increment *N* by 1 and use the next slot.

The end-device can now compute the real preamble length:

$$\boldsymbol{PreambleLength} = \max\left(8;\ floor\left(\frac{DriftError}{T_{SYMB}}\right) + 1 + 6 + \mathrm{CadToRx}\right)$$

> **Note:** The *PreambleLength* takes into account the minimum number of symbols required by the relay to detect and receive the WOR frame.

The previous formula gives the start time ($\boldsymbol{T_{START}}$) and preamble length ($\boldsymbol{PreambleLength}$) for the last reference channel. If the end-device wants to use the other channel, it has to add/subtract half of `CADPeriodicity` to/from T_{NEXT}.

Every time the end-device transmits a WOR Class A Uplink frame it must save the following variables:

- T_{LAST}: The value of the device's internal clock counter at the beginning of the WOR transmission (in milliseconds)
- *PreambleLength* of WOR

Every time the end-device receives a valid wake-up WOR ACK, it SHALL:

- Save the channel used as the last reference channel
- Update T_{REF} by applying the following formula:

$$T_{REF} = T_{LAST} + PreambleLength * T_{SYMB} - T_{OFFSET}$$

Every time a relay receives a valid WOR frame, it has to compute T_{OFFSET}:

$$T_{OFFSET} = ceil(T_{END} - T_{SCAN} - TOA(WOR) + (12 + 4.25) * T_{SYMB})$$

With:

- T_{SYMB}: Symbol time
- T_{END}: End of the reception of the WOR frame
- T_{SCAN}: Start of the channel scan
- TOA(WOR): Time on air of the WOR frame

> **Note:** 16.25 symbols are added to only have the length of the detected preamble without the payload and sync word.

Example:

An end-device in the **initialized** state transmits a WOR Class A Uplink frame at T_{LAST} = 1234 ms on the default channel (SF10 BW125). The crystal accuracy of the end-device is 20ppm. The preamble length of this WOR frame is 133 symbols.

The relay scans this channel every 500 ms. The WOR frame is detected at T_{SCAN} = 87 654 ms (different time base). The reception of the WOR frame is completed at T_{END} = 88 734 ms.

$$T_{OFFSET} = ceil(88734 - 87654 - 321.536 + (12 + 4.25) * 8.192) = 892\ ms$$

The end-device receives the WOR ACK frame, which contains the following fields:

- `TOffset`: 892 ms
- `CADPeriodicity`: 500 ms
- `XTALAccuracy`: 30 ppm
- `CadToRx`: 4 symbols

The end-device is now **synchronized** and can compute the T_{REF} value:

$$T_{REF} = 1234 + 133 * 8.192 - 892 = 1\ 431\ ms$$

At T_{NOW} = 61 000 ms (~1 minute later), the end-device wants to transmit a new WOR Class A Uplink frame. Because it is now in the **synchronized** state, it will compute the T_{NEXT} and *PreambleLength*:

$$\boldsymbol{T_{NEXT}} = ceil\left(\frac{61000 - 1431}{500}\right) * 500 + 1431 = 61\ 431\ ms$$

$$\boldsymbol{DriftErrorMs} = \frac{(20 + 30) * (61431 - 431)}{1000000} = 3\ ms$$

$$\boldsymbol{T_{START}} = 61431 - \frac{3}{2} = 61\ 430\ ms$$

$$\boldsymbol{PreambleLength} = \max\left(8;\ floor\left(\frac{3}{8.192}\right) + 1 + 6 + 4\right) = 12\ symbols$$

For some reason, the end-device does not receive the ACK.

At T_{NOW} = 3 600 400 ms (~1 hour later), the end-device wants to transmit a new WOR Class A Uplink frame.

$$\boldsymbol{T_{NEXT}} = ceil\left(\frac{3600400 - 1431}{500}\right) * 500 + \ 1431 = 3\,600\,431\ ms$$

$$\boldsymbol{DriftErrorMs} = \frac{(20 + 30) * (3600431 - 1431)}{1000000} = 180\ ms$$

$$\boldsymbol{T_{START}} = 36000431 - \frac{180}{2} = 3\,600\,341 ms$$

This T_{START} value is in the “past” so we will use the next slot at T_{NEXT} = 3 600 931 and start the transmission at T_{START} = 3 600 841 ms.

$$\boldsymbol{PreambleLength} = \max\left(8;\ floor\left(\frac{180}{8.192}\right) + 1 + 6 + 4\right) = 32\ symbols$$

For some reason, the end-device does not receive the ACK.

At T_{NOW} = 36 000 000 ms (~10 hour later), the end-device wants to transmit a new WOR Class A Uplink frame.

$$\boldsymbol{T_{NEXT}} = ceil\left(\frac{36000000 - 1431}{500}\right) * 500 + 1431 = 36\,000\,431\ ms$$

$$\boldsymbol{DriftErrorMs} = \frac{(20 + 30) * (36000431 - 1431)}{1000000} = 1\,800\ ms$$

The drift error is bigger than the `CADPeriodicity`, so the end-device will go to the **unsynchronized** state, use a preamble of 72 symbols (because `CADPeriodicity` is 500ms), and start the transmission of the WOR frame when it wants.

APPENDIX 2: SECURITY CONSIDERATIONS

This Appendix highlights potential security issues of the relayed network architecture per this specification.

The purpose is to understand these issues, in order to make a decision whether to accept them or design remedies for them.

Elements that are new in the relay architecture, compared to the vanilla LoRaWAN architecture, are:

- The relay node itself
- The relay protocol between the relay and the NS (data forwarding and management)
- The relay protocol between the end-device and the NS (management)
- The WOR protocol and WOR link between the end-device and the relay
- The LoRaWAN link between the end-device and relay, including the new RXR window

The elements of the relayed network architecture are shown in Figure 16, with the new elements in red.

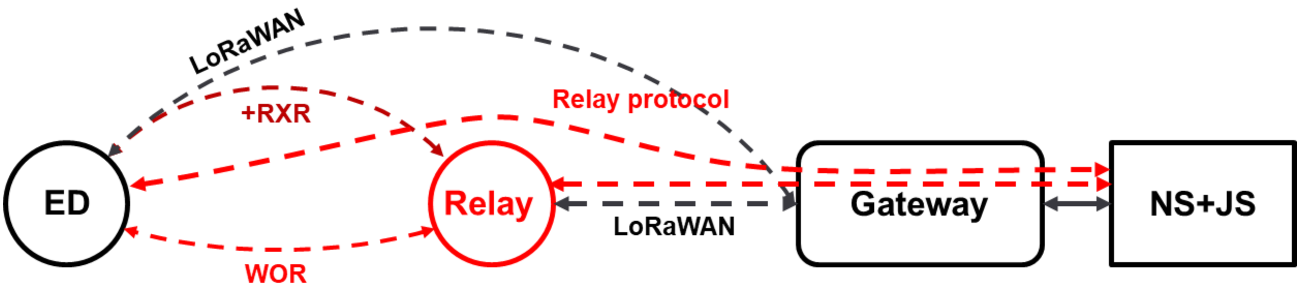

Figure 16: Relayed network architecture

The following sections discuss the security issues potentially associated with these new elements.

A. Relay Node and Associated Relay Protocol

The relay node introduces an extra point of attack in the overall architecture. The relay is likely to be fairly easy to reach physically and is unlikely to contain tamper-proof hardware, therefore:

- The relay could be removed or destroyed, however disrupting its service will only affect a limited number of end-devices (in the order of 10).
- The relay could be compromised, like any end-device.

First, a compromised relay could reveal the keys stored in it and used to secure the communication between the relay and the end-devices it serves. They are sessions keys, which can be refreshed by an end-device Join/Rejoin. And, the number of end-devices served by a relay is expected to be in the order of 10, which limits the damage.

Recommendation: The NS should send the relay only the keys for end-devices that the relay is highly likely to serve in the near future.

Second, the relay management protocol exposes a new surface of attack to the NS, which a compromised relay could try to exploit:

- In the data plane (forwarded uplinks): A compromised relay could send a flow of spoofed forwarded uplinks. The encapsulated payloads are MIC'ed (see Section 8.1) and will be verified as invalid and dropped by the NS. There is no amplification effect. The rate is limited by the LoRaWAN uplink from the relay to the network. This is no different than a compromised end-device in direct connection with the network, sending traffic to an application server, except that in our case, the application server is the NS.
- In the control plane (MAC commands, see Table 30):
    - The ***NotifyNewEndDeviceReq*** MAC command may trigger activity in the Network Server/Join Server (JS) to search for the purported new end-device's record, to make a decision whether relaying should be enabled to that end-device and potentially to compute keys for it. The rate of such events is limited by the LoRaWAN uplink data rate from the relay to the network. This rate is no higher than that of a fake end-device sending Join-Requests. However, implementers might consider providing extra rate limitations in the NS for the ***NotifyNewEndDeviceReq*** MAC command, per relay.
    - The other relay MAC commands only have a local effect in the NS, since they only relate to the relay state by itself.
- Through potential protocol implementation bugs on the NS side. This is not specific to the relaying protocol.

*Recommendation: Evaluate the resource consumed by the **NotifyNewEndDeviceReq** MAC command in an NS and consider implementing extra rate limitations in the NS for the **NotifyNewEndDeviceReq** command, per relay*.

B. Relay Protocol Between the End-Device and NS

This protocol consists of just one MAC command in each direction for the NS to configure the relaying parameters in the end-device, and for the end-device to acknowledge the change (see Section 10.2).

These MAC commands are either exchanged directly between the end-device and the NS or forwarded by a relay. Even in the latter case, the relay cannot tamper with the forwarded command since it does not have the "end-device to NS" session keys.

The possible attacks are therefore no different from that on any MAC command of any end-device.

Furthermore, the content exchanged between the end-device and the NS through the relay protocol (i.e., end-device additional WOR channel configuration) is not particularly sensitive.

C. WOR Protocol

The WOR protocol consists of WOR Join-Request frames and of the WOR Class A Uplink / WOR ACK frame pairs.

WOR CLASS A UPLINK AND WOR ACK FRAMES

The WOR Class A Uplink and WOR ACK frames are encrypted and MIC'ed with network session keys (see Sections 5.3.2 and 6.2), much like regular LoRaWAN frames in a regular LoRaWAN network. However, one notable difference is that the "end-device to relay" session keys (actually, `RootWorSKey`, a precursor of them, see Section 4) are sent to the relay by the NS (over the relay's LoRaWAN link). How secure should the derivation and the transport of the session keys to the relay be? Which begs the question: what are the consequences of an attacker being able to decrypt or forge WOR frames?

If `RootWorSKey` is guessed or revealed, an attacker can then easily compute the `WorSEncKey` and `WorSIntKey` if it also knows the `DevAddr` of the associated end-device.

If a `WorSEncKey` is guessed or revealed, an attacker can:

- Decrypt the WOR Relay Class A Uplink message, thereby learning the physical parameters of the forthcoming Class A Uplink. This allows mounting a more selective jamming attack of the channel from the end-device to the relay.
- Decrypt the WOR ACK, thereby learning some relay internal states (e.g., data rate, timing, or forwarding status). This allows mounting a more selective jamming attack of the channel from the relay to the end-device.

If a `WorSIntKey` is guessed or revealed, an attacker can:

- Spoof a valid WOR Relay Class A Uplink message. This fools the relay into receiving and forwarding a forthcoming Class A Uplink frame (which constitutes a DoS attack against the relay, a radio amplification attack, and potentially a hijacking of the relay function for a private LoRa network).
- Spoof a WOR ACK message. This lets an end-device believe that a relay is available to forward its messages, while the relay has actually vanished (blackhole attack). It also allows the attacker to advertise the wrong timing, wrong data rate, or wrong crystal accuracy (DoS attack against the end-device).

Decision: The transport of `RootWorSKey` does not warrant a Key-Encryption-Key mechanism; transporting `RootWorSKey` directly in the payload of a regular LoRaWAN frame from the NS to the relay is deemed secure enough.

Another attention point is that the WOR Class A Uplink is not cryptographically linked to the forthcoming LoRaWAN uplink. As a consequence, a relay that has received a valid WOR Class A Uplink frame will happily forward any LoRaWAN frame that it receives at the appropriate time, with the physical parameters listed in the WOR Class A Uplink, and with the right `DevAddr`. That LoRaWAN frame could be one forged by an attacker. This attack provides no benefit to the attacker, beyond the energy deprivation or the DoS attack on the relay/network.

Decision: No cryptographic link between the WOR and the forthcoming LoRaWAN uplink is established.

WOR Relay Join-Requests

The WOR Relay Join-Request is not authenticated by the relay. The rationale is that the end-device sending the WOR Relay Join-Request may have never been associated with the relay or even with the network before, so there is nothing in the relay to authenticate the WOR Relay Join-Request with. Trusting the relay with a root key (i.e., AppKey in 1.0.x or NwkKey in 1.1.x or higher) is not acceptable, per the requirements document.

As a consequence of the WOR Relay Join-Requests not being authenticated at the relay, the relay will forward any forthcoming Join-Request, and it will allocate resources (e.g., memory, time on air) and spend energy for this forwarding. This leads to several sorts of attacks:

- Denial of Service: The resources allocated to the malicious Join-Request may prevent the relay from serving concurrent legitimate requests.
- Deprivation of Energy: The relay will spend energy serving the malicious Join-Requests, leading to its earlier death by energy deprivation (if energy constrained). This in turn leads to a deferred Denial of Service attack.
- Hijacking of the relay for the attacker's personal use: If physical forwarding (a.k.a. impersonation) is used for the forwarding, the malicious Join-Requests are repeated, unchanged. The relays can therefore be abused by an attacker to forward proprietary messages, encoded into LoRaWAN Join-Request frames, between its own nodes. If encapsulation is used for forwarding, the forwarded malicious Join-Request is rendered opaque to the attacker by the relay-to-NS encryption. Therefore the hijacking provides no benefit to the attacker (the relay should check the length of the Join-Request before forwarding it, so that the forwarded frame length cannot be exploited).

All of these attacks can be mitigated, but not eliminated, by filtering and rate limiting.

- The relay should check that the frame to be forwarded is a Join-Request, as much as it can tell (header, length, etc,).
- Filtering by `DevEUI` reduces the set of `DevEUI`s for which Join-Requests are forwarded. However, a brute force attack may discover (part of) that set. Filtering by `DevEUI` assumes that the relay is preconfigured to only accept a set of end-devices, which is a use case limitation.
- Rate limiting per `DevEUI` is expensive to implement and not efficient if a wide range of forwarded `DevEUI`s is offered to the attacker.
- Global rate limiting at the relay level is easy to implement but results in denial of service to legitimate Join-Requests.

Recommendation: The relay should do some sanity checking on the Join-Request before considering whether to forward it, thereby saving on the credits of the rate limiting function (see Section 8.8).

D. New Links

The new links are susceptible to jamming, as with any radio link.

The new links are shown in Figure 17, with the receive ends in red.

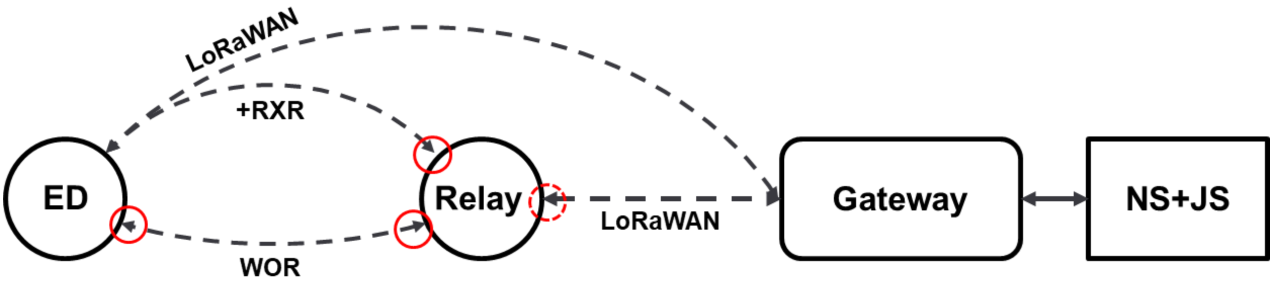

Figure 17: New radio links

There are four new radio interfaces:

- WOR link, end-device side
- WOR link, relay side
- LoRaWAN link between the end-device and relay, relay side
- LoRaWAN link between the relay and gateways, relay side

Jamming of the last one is no different than jamming an end-device in the current LoRaWAN architecture, except that more end-devices are affected. However, their number is limited. The other three interfaces are discussed in subsections below.

Jamming of the WOR Channel at the End-device

Because the relay is meant to use end-device-class radio chips, the WOR link between the end-device and the relay uses one or two physical channels (frequency/data rate) only (see Section 0).

Compared to the wider range of frequencies and physical parameters made available to the link between an end-device and the network gateways, this leads to easier jamming by an attacker (e.g., continuous narrow-band signal).

This susceptibility to jamming is mitigated if two WOR channels are used, which comes at the expense of more energy consumed at the relay. This mitigation is only effective after the end-device has acquired the knowledge about the additional channel.

After the WOR channel(s) used by a relay have been discovered by an attacker, a jammer located close to or focused toward the end-device can easily disrupt the link from the relay to the end-device, thereby preventing the end-device from receiving the WOR ACK.

The consequences of the end-device being unable to receive the WOR ACK are:

- The end-device will not acquire or maintain synchronization to the relay. Therefore, it will send unnecessarily long preambles in forthcoming transmissions.
- The end-device will keep retrying to reach the relay, spending resources, and eventually fail to discover the existence of the relay.
- The end-device may refrain from sending the forthcoming LoRaWAN uplink, thereby missing the opportunity to have the uplink received by the relay, and potentially also by a gateway.

This attack only disrupts the operation of one end-device.

Recommendation: Have the end-device send the Class A Uplink anyway from time to time, even when it has not received the WOR ACK. In the attack case described here, this mitigation may allow the Class A Uplink to be received and forwarded by the relay, and maybe even to be directly received by a gateway. This is allowed by this specification (see Section 3.4) and configurable by the NS into each end-device (see Section 10.2).

Jamming of the WOR Channel at the Relay

After the WOR channel(s) used by a relay have been discovered by an attacker, a jammer located close to or focused toward the relay can easily disrupt the link from the relay to the end-device, thereby preventing the relay from receiving the WOR Relay Join-Request or the WOR Relay Class A Uplink.

The consequences of the relay being unable to receive the WOR Relay Class A Uplink are:

- The relay will not listen for the forthcoming Class A Uplink transmission of the end-device and therefore will not be able to forward it.
- The end-device will not acquire or maintain synchronization to the relay. Therefore it will send unnecessarily long preambles in future transmissions.
- The end-device will keep retrying to reach the relay, spending resources, and eventually failing to discover the existence of the relay.

- The end-device may refrain from sending the forthcoming Class A Uplink, thereby missing the opportunity to have the uplink potentially received by a gateway.

This type of attack disrupts the traffic of all end-devices served by that relay.

The last consequence is mitigated by having the end-devices send the Class A Uplink anyway from time to time, even when they have not received the WOR ACK. In the attack case described here, this mitigation may allow the Class A Uplink to be received by a gateway.

Recommendations:

- *Activate the second WOR channel in the relay if jamming is a concern.*
- *The end-device should alternate between WOR channels if a WOR ACK is not received.*
- *Have the end-device send the Class A Uplink anyway from time to time, even when it has not received the WOR ACK. This is allowed by this specification (see Section 3.4) and configurable by the NS into each end-device (see Section 10.2).*

JAMMING OF THE LORAWAN LINK BETWEEN THE END-DEVICE AND THE RELAY, AT THE RELAY SIDE

Wideband jamming close to the relay will disable its reception. This will prevent it from serving the end-devices attached to it. The latter are expected to be in small numbers.

A more selective jamming can be mounted by an attacker able to eavesdrop on the WOR link; it learns the physical parameters of the upcoming LoRaWAN transmission from the WOR link.

Both attacks are more difficult to mount than jamming the WOR link at the relay side and provide no additional benefit.

Recommendation: The relay should be able to detect and report severe wideband jamming to the NS, maybe upon request by the NS.

APPENDIX 3: FILTER/FORWARD EXAMPLE

The following table represents the forward/filter lookup table of a relay for Join-Request:

#	JoinEUI:DevEUI	Size [bytes]	Rules
0	--	0	Filter
1	0xAB 0xCD 0xEF	3	Forward
2	0xAB 0xCD 0xEF 0xAB 0xCD 0xEF 0xAB 0xCD	8	Filter
3	0xAB 0xCD 0xEF 0xAB 0xCD 0xEF 0xAB 0xCD - 0x12 0x34 0x56 0x78 0x28 0x37 0x46	15	Forward
4	0xAB 0xCD 0xEF 0xAB 0xCD 0xEF 0xAB 0xCD - 0x12 0x34 0x56 0x78 0x28 0x37 0x46 0x48	16	Filter

The following table represents a list of end-devices:

#	Join EUI	DevEUI
0	0xAB 0xCD 0xEF 0xAB 0xCD 0xEF 0xAB 0xCD	0x12 0x34 0x56 0x78 0x28 0x37 0x40 0x44
1	0xAB 0xCD 0xEF 0xAB 0xCD 0xEF 0xAB 0xAF	0x12 0x34 0x56 0x78 0x28 0x37 0x46 0x45
2	0xAB 0xCD 0xEF 0xAB 0xCD 0xEF 0xAB 0xCD	0x12 0x34 0x56 0x78 0x28 0x37 0x46 0x46
3	0xAB 0xCD 0xEF 0xAB 0xCD 0xEF 0xAB 0xCD	0x12 0x34 0x56 0x78 0x28 0x37 0x46 0x47
4	0xAB 0xCD 0xEF 0xAB 0xCD 0xEF 0xAB 0xCD	0x12 0x34 0x56 0x78 0x28 0x37 0x46 0x48
5	0x91 0x82 0x73 0x64 0x55 0xEF 0xAB 0xCD	0x12 0x34 0x56 0x78 0x28 0x37 0x46 0x49

Where:

- ED0 is filtered (rule 2)
- ED1 is forwarded (rule 1)
- ED2 and ED3 are forwarded (rule 3)
- ED4 is filtered (rule 4)
- ED5 is filtered (rule 0)

Notes:

- Rule 0 has been set to change the default behavior of the relay to filter unknown end-devices.
- Rule 1 has been set to forward all end-devices of a specific OUI (Organizationally Unique Identifier).
- Rule 2 has been set to filter a specific `JoinEUI`.
- Rule 3 has been set to forward a specific range of `DevEUIs` within a specific `JoinEUI`.
- Rule 4 has been set to filter a specific `DevEUI` in the previous set of `DevEUIs` (refer to rule 3)

APPENDIX 4: NETWORK TIME SYNCHRONIZATION

An end-device can send the command ***DeviceTimeReq*** to receive the network date and time from the Network Server. The time provided is the network time captured at the end of the uplink transmission.

When a relay is added in the loop, the Network Server has to perform some calculations to find the time of emission of the original uplink.

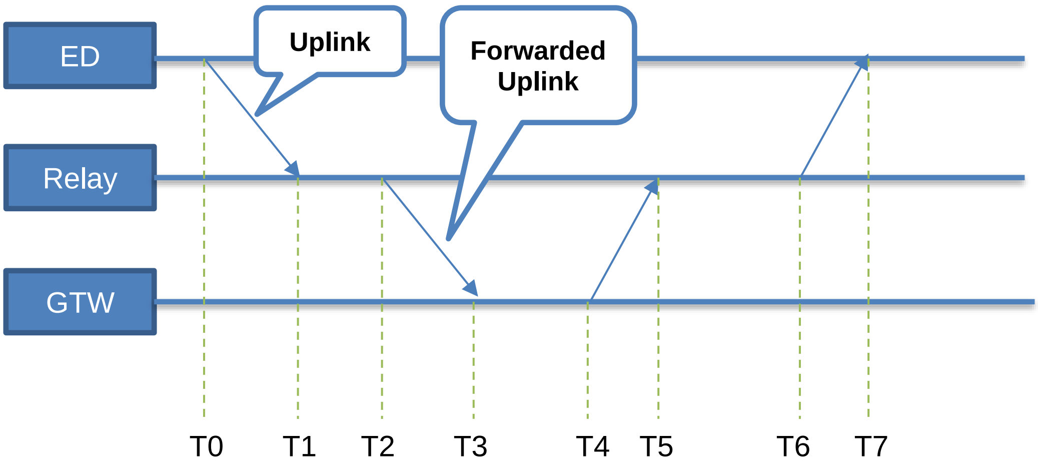

Figure 18: Network time synchronization

The Network Server is able to compute the time on air of the forwarded message from the relay:

T2 = T3 – TOA (forwarded uplink)

This specification mandates that the delay between the reception of the uplink from the end-device and its forwarding by the relay is constant, its value is known as RELAY_FWD_DELAY.

T1 = T2 - RELAY_FWD_DELAY

The Network Server is now able to send a ***DeviceTimeAns*** with a good correction value:

T1 = T3 – TOA (forwarded message) - RELAY_FWD_DELAY

APPENDIX 5: END-DEVICE RELAY MODE ACTIVATION MANAGEMENT

As mentioned in this specification, prior to joining a LoRaWAN network, an end-device should autonomously decide whether or not to enable relay mode.

The following recommendation only applies to an end-device in "end-device controlled" mode (opposed to network controlled - Refer to "Table 40: RelayModeActivation format", value 0, 1 or 2).

When the end-device sends a Join-Request, it is recommended to enable the relay mode once every 4 frames. If the Join-Accept has been received on RXR, the end-device should keep the relay mode enabled. Otherwise, it should disable the relay mode.

Once the end-device is connected to the network:

- It should disable the relay mode if it does not receive a WOR ACK after 8 WORs. It should enable the relay mode if it does not receive a valid downlink after 16 uplinks.